A WEB-BASED NEGOTIATION SERVER FOR SUPPORTING ELECTRONIC COMMERCE

By

CHUNBO HUANG

A DISSERTATION PRESENTED TO THE GRADUATE SCHOOL
OF THE UNIVERSITY OF FLORIDA IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

UNIVERSITY OF FLORIDA

2000

Copyright 2000

By

Chunbo Huang

Dedicated to my dear parents and wife

ACKNOWLEDGMENTS

First and foremost, I would like to express my gratitude towards my advisor, Dr. Stanley Y.W. Su, for giving me the opportunity to work with him in the Database Systems Research and Development Center. His excellent advice, great guidance and fruitful discussions on many aspects of this dissertation are invaluable. My great appreciation goes to Dr. Joachim Hammer, my co-advisor, for constantly providing me with valuable comments and suggestions during the dissertation work. I wish to thank Dr. Herman Lam, Dr. Abdelsalam Helal, and Dr. Antonio A. Arroyo for serving on my supervisory committee and for taking time to read and comment on my dissertation.

I am grateful to Dr. Yihua Huang, a visiting professor from Nanjing University, China, for his suggestions and help on my research. I also would like to thank all the graduate students participating in the implementation task of this research. They are Haifei Li, Minsoo Lee, Charnyote Pluempitiwiriyawej, Youzhong Liu, and Jie Meng. I can not forget to thank Ms. Sharon Grant who is always willing to help in any way she can.

Special thanks and love go to my wife and my parents. Without their help and encouragement, this work would have been impossible to complete.

This project is supported by IBM (contract number: CLT-H98014-00), which is the main contractor of a project funded by the Advanced Technology Program of the National Institute of Standards and Technology.

TABLE OF CONTENT

ACKNOWLEDGEMENTS iv

ABSTRACT vii

CHAPTERS

1 INTRODUCTION 1

2 SURVEY OF RELATED WORK 7

2.1 Negotiation Protocol 7
2.2 Game Theory 11
2.3 Negotiation Support Systems 12
2.4 Agent Technologies 13
2.5 Machine Learning 15
2.6 Constraint Satisfaction Problem 17
2.7 Active Database and Rules 19
2.8 XML and BOD 20

3 WEB-BASED AUTOMATED NEGOTIATION SERVER OVERVIEW 24

3.1 System Architecture 24
3.2 Automated Negotiation Process Overview 27

4 NEGOTIATION KNOWLEDGE ACQUISITION AND REPRESENTATION 31

4.1 Object-oriented Constraint-Based Requirement Specification 31
4.2 Rule-based Negotiation Strategy Specification 38
4.3 Cost Benefit Analysis Model for Proposal Evaluation 44

5 AUTOMATED NEGOTIATION PROCESS 50

5.1 Negotiation Primitives and Negotiation Protocol 51
5.2 Message Content Specification, Formatting, and Transmission 64
5.3 Negotiation Decision Process 66
5.4 Negotiation Constraint Satisfaction Processing 71
5.5 Event-Trigger-Rule Server 77
5.6 Example Scenarios 79

6 SYSTEM DESIGN AND IMPLEMENTATION 81

6.1 System Overview 81
6.2 Enhance the Usability of Negotiation Server 86
6.3 Integration Testing of the Negotiation Server in a Supply Chain Management Scenario 90

7 CONCLUSION AND FUTURE RESEARCH 93

7.1 Conclusion 93
7.2 Future Research 95

APPENDICES

A BNF SYNTAX OF OBJECT-ORIENTED CONSTRAINT SPECIFICATION LANGUAGE (OOCSL) 98

B PROPOSE PROPOSAL BOD XML DATA TYPE DEFINITION 100

C PROPOSE PROPOSAL BOD XML MESSAGE 101

D GENERATED NEGOTIATION STRATEGY RULE CODE IN JAVA 103

LIST OF REFERENCES 105

BIOGRAPHICAL SKETCH 113

Abstract of Dissertation Presented to the Graduate School
of the University of Florida in Partial Fulfillment of the
Requirements for the Degree of Doctor of Philosophy

A WEB-BASED NEGOTIATION SERVER
FOR SUPPORTING ELECTRONIC COMMERCE

By

Chunbo Huang

May 2000

Chairman: Dr. Stanley Su
Cochairman: Dr. Joachim Hammer
Major Department: Computer and Information Science and Engineering

Automating negotiation as much as possible is a very important and challenging research problem. Solving this problem can benefit several areas of interest: electronic commerce, supply chain management, manufacturing resource planning and scheduling, distributed artificial intelligence, and multi-agent collaborative systems. This dissertation presents methodologies and techniques needed for conducting automated negotiations via the Internet. A replicable negotiation server has been developed, which can be installed at multiple sites, such as Web servers, to conduct automated negotiations on behalf of users and/or business organizations. The negotiation process is carried out between two negotiation servers in a bilateral bargaining situation based on a set of negotiation primitives, which, together with product requirements and constraint specifications, are transmitted between the servers as business object documents by following a formalized

negotiation protocol. To enhance the negotiation servers with effective and intelligent negotiation capabilities, human negotiation knowledge is incorporated in the negotiation process. Different from existing approaches in which human knowledge is hard-coded in negotiation agent programs, we use (1) a high-level object-oriented constraint specification language for representing requirement and constraint specifications, (2) an Event-Trigger-Rule server for implementing negotiation strategies which are formulated as rules, and (3) a cost-benefit analysis and decision model for the evaluation and selection of negotiation alternatives under consideration. A number of GUI tools have been developed to facilitate the specification of product requirements and constraints, negotiation strategies, preference scoring and aggregation for cost-benefit analysis. Run-time consoles are also available for controlling and monitoring of concurrent negotiation processes as well as for user intervention. Replicated negotiation servers together with their supporting system components have been integrated with several commercial products to form an automated supply chain management system.

CHAPTER 1
INTRODUCTION

Negotiation is a widely accepted practice when conducting business. According to Pruitt [PRU81], negotiation can be defined as "the process by which a joint decision is made by two or more parties. The parties first verbalize contradictory demands and then move towards agreement by a process of concession making or search for new alternatives." With the advancement of computer and information technologies, automated negotiation has become an important research topic in several areas: Supply Chain Management (SCM), Manufacturing and Resource Planning (MRP), Distributed Artificial Intelligence (DAI), and Multi-Agent System (MAS). DAI and MAS use negotiation to coordinate intelligent agents to complete designated tasks. MRP uses negotiation as a dynamic and efficient mechanism to generate and execute manufacturing plans. SCM uses negotiation to establish a dynamic and profitable coalition for resource supplying and product delivery.

Another main application area that can benefit from automated negotiation is Internet-based Electronic Commerce (E-Commerce). In complex, non-electronic business transactions, individual consumers and companies want to negotiate the price, the delivery day, the quality of goods and services, and other purchase conditions. Traditionally, negotiations are conducted by people involved in business transactions.

In the Internet-based E-Commerce paradigm, companies publish their catalogs on the Internet for browsing, receive purchase orders from consumers or business partners, negotiate the terms of the deals, order the delivery of goods and/or services, accept payments, and support customers, in an all-electronics fashion. According to Sokol [SOK89], "Electronic Commerce is the sharing of information using a wide variety of different electronic technologies, between organizations doing business with one another...[it includes] procedures, policies, and strategies to support incorporation of these electronic messages into the business environment." With the blossoming of the Internet and the World Wide Web (WWW) in the past five years, E-Commerce has become a new business paradigm that has attracted worldwide attention. Today, there are more than 30 million Internet users, and the number is projected to grow to 90 million in the near future. Since these users are potential consumers, there are incentives for companies, ranging from multinational corporations to small mom-and-pop operations, to conduct their business through the Internet. It has been predicted that by 2000, 80 percent of Fortune 500 companies will conduct their core business through the Internet. E-Commerce related revenue reached $245 billion in 1994 and it is projected to increase to $1.65 trillion in 2000 and $3 trillion in 2005 [TYG98].

In both consumer-to-business and business-to-business E-Commerce, it is desirable to carry out this negotiation process automatically or at least semi-automatically with human intervention when necessary. The objective of automated negotiation is to replace or assist humans by providing software that can automatically carry out the negotiation process and thereby achieve faster execution speed, higher consistency, and significant

reduction of human errors. To automate the negotiation process for Internet based E-Commerce systems several important issues must be addressed:

(1) Acquire and represent negotiation knowledge. To enable automated negotiation systems or assist a human to conduct effective negotiations, automated negotiation systems must be intelligent. For instance, they must have the information and knowledge about people or business organizations' negotiation requirements, preferences for different alternatives, and negotiation strategies, tactics, and business policies. This kind of information can be provided by human negotiation experts. Therefore, the methodologies to represent the knowledge, use the knowledge efficiently and effectively, and maintain and modify the knowledge when necessary are also needed for conducting automated negotiations.

(2) Automate the negotiation decision process. A human-based negotiation decision process arbitrarily proceeds under the influence of time, location, persons involved, regulations, conventions, etc. It is difficult to automatically conduct negotiation by a computer system without formally analyzing the negotiation decision process. Specifically, we need to analyze the decision process that governs how the negotiation system decides to accept, reject proposals, or generates counterproposal. We also need to determine what kind of information is needed to support the decision-making process.

(3) Achieve effective interaction between negotiation systems. Traditionally, interactions between negotiation parties are through natural languages. A natural language has a high expressive power, but it is highly ambiguous and difficult to be processed by computers. Thus, formal negotiation primitives, negotiation protocol, and message specifications are needed

(4) Design a suitable system architecture. To take full advantage of the Internet as a public resource and infrastructure, the negotiation system architecture must be designed based on the Internet and must be scalable to support concurrent use of negotiation services by Internet users in a highly distributed environment. Ease of use is another important issue for application-oriented systems, including easy installation and configuration of the system. Moreover, in real business applications, it is often unrealistic to completely delegate the negotiation task to a computer software. Instead, human interactions may be required from time to time. Therefore, graphic user interfaces are essential components of an automated negotiation system. They can be used for human to control and monitor the negotiation process.

To date, existing efforts have not effectively solved the above problems. This fact motivated our research to develop methodologies and techniques capable of achieving automated negotiations across the Internet. We try to develop a replicable negotiation server which can be installed at multiple sites, similar to Web servers, to conduct automated negotiations on behalf of users and/or business organizations. The negotiation process is carried out between two negotiation servers in a bilateral bargaining situation. It is based on a set of negotiation primitives, which, together with product requirements and constraint specifications, are transmitted between the servers as business object documents by following a formalized negotiation protocol. To enhance the negotiation servers with effective and intelligent negotiation capabilities, human negotiation knowledge is incorporated into the negotiation process. Contrary to existing approaches in which human knowledge is hard-coded in the form of negotiation agent programs, we

use (1) a high-level object-oriented constraint specification language for requirement and constraint specifications, (2) an Event-Trigger-Rule server for implementing negotiation strategies which are formulated as rules, and (3) a cost-benefit analysis and decision model for the evaluation and selection of negotiation alternatives under consideration. A number of GUI tools have been developed to facilitate the specification of product requirements and constraints, negotiation strategies, preference scoring and aggregation for cost-benefit analysis. At run time, an efficient constraint satisfaction processing (CSP) engine is provided so that the constraints defined in the proposal and the registered requirements can be evaluated in a more efficient and effective way than the traditional CSP engine. The result of the CSP evaluation serves as the basis for the negotiation system to trigger the corresponding negotiation strategic rules (in the case of violations) or to make the proper selection from multiple alternatives that satisfied the constraint evaluation. Run-time consoles are also available for controlling and monitoring concurrent negotiation processes as well as for user interventions. Replicated negotiation servers together with their supporting system components have been integrated with several commercial products to form a complex automated supply chain management system.

The design and development of the negotiation server documented in this dissertation is the result of a group effort. The author's main contribution is in this effort are: 1) the design of the system architecture, 2) the development of a constraint-based, object-oriented content specification language for negotiation requirement and constraint specifications, 3) the design and implementation of the constraint satisfaction processor,

4) the design and specification of negotiation primitives and protocol, and 5) the integration with components and GUIs implemented by other group members.

The remainder of this dissertation is organized as follows: Chapter 2 surveys the related research work. Chapter 3 provides an overview of our web-based negotiation server architecture and the negotiation process. Our solutions for the negotiation knowledge acquisition and representation are described in Chapter 4. We present the key techniques to support automated negotiation process execution in Chapter 5. In Chapter 6, the prototype system design and implementation is presented. Finally, our conclusions and plans for future research are summarized in Chapter 7.

CHAPTER 2
SURVEY OF RELATED WORK

Understanding and automating negotiation is a broad and complex problem which has attracted researchers from many diverse areas such as communication, linguistics, politics, diplomacy, game theory, economics, and computer science. Given the recent interest in automated negotiation, a considerable amount of research work on automated negotiation systems has become available. In the following sections, we give a survey about related work in negotiation protocol, game theory, negotiation support systems, agent technologies, and machine learning. We also survey the related techniques used in this dissertation to implement a replicable Web-based automated negotiation server, including Constraint Satisfaction Problem solving, Event-Condition-Action (ECA) rules used in active database systems, eXtensible Markup Languages (XML) and Business Object Document (BOD).

2.1 Negotiation Protocol

Due to the long history and wide applicability of human-based negotiation, there exist multiple types of negotiation protocols. Bidding, in which a buyer specifies the product or service he wants to acquire from suppliers and asks for their bids, is the simplest form of negotiation. Based on the bids, the buyer selects the supplier(s) from whom to order the goods or services. The Contract Net Protocol (CNP) of Davis and Smith [SMI80] is a good example of bidding. Auction, in which the server (normally,

the seller) initiates an auction and monitors the auction process while the bidding buyers send their own purchasing conditions to the server, is another form of negotiation. The server follows a certain auction protocol to pick up the final trading partner from the participating buyers. Several different auction protocols exist, e.g., English auction, Dutch auction, first-price & sealed-bid auction, Vickrey auction, double auction, etc. In the English auction, the auctioneer (server) begins at the seller's reservation price, and solicits progressively higher bids from the participants until only one bidder remains. In the Dutch auction, the auctioneer begins with the highest price and progressively lowers the price until one bidder accepts the bid. Other types of auctions also have different auctioning rules [KUM98, LEE97]

Auction has been widely used in E-Commerce applications on the Internet. Beam [BEA97a] discusses the difference between the Internet-based automated auction and the traditional auction. For example, the auction time of an Internet-based auction can last longer since the participants do not have to physically gather at the auction site during the auction process. Internet-based auctions should also support multiple current auction sessions including a large number of Internet users to participate in the auctions. Many commercial auction sites have been running: Onsale [ONS99], Ebay [EBA99], etc. Some popular Internet sites also provide auction services: Yahoo [YAH99] and Amazon.com [AMA99] are a couple of examples. However, the technologies used by the current commercial auction servers are insufficient to satisfy the fast-growing auction activity on the Internet. First, although different auction servers have different protocols, the protocol is fixed for a particular auction server and can not be configured according to different time, products, or auctioneers. This situation may prevent the auction site from

expanding the auction services to other products or other group of Internet users. Second, although the auction server has been automated, the bidder interface is completely based on human input, which may take human bidders a long time to complete a single auction session. Considering the large number of Internet bidders, current system architectures are inefficient.

To overcome the existing shortcomings, the Unviersity of Michigan has developed a sophisticated auction prototype system, AuctionBot [WUR99, WUR98, AUC99]. In AuctionBot, users create new auctions to buy or sell products by choosing from a selection of auction protocols and specifying parameters, e.g., clearing times, the method for resolving bidding ties, the number of sellers permitted, etc. Buyers and sellers can then bid according to the specified auction protocol. In a typical scenario, after entering an auction, a seller would bid a reservation price and let AuctionBot manage and enforce buyer's bidding according to the specified auction protocols and parameters. AuctionBot also provides application programmable interfaces (APIs) allowing users to create personalized software agents to autonomously compete as bidders in the AuctionBot marketplace. The bidders are responsible for encoding their own bidding strategies in the software agents.

However, like other commercial auction servers, AuctionBot still focuses on the single-issue (in their case, price) negotiation. In [GUT98] the authors argue that multi-issue/multi-attribute auctions based on value comparisons, such as delivery time, customer service, product quality, and brand name, is a better approach but requires more complex auction protocols and bidding strategies. A prototype system supporting multi-attribute auction has been reported in [BIC99].

A more complex negotiation protocol is bargaining in which a negotiation proposal and possible subsequent counterproposals between negotiation parties are exchanged until a mutual agreement or disagreement is reached. The study of the bargaining type of negotiation has long attracted economists because it is fundamental to exchanges and markets. General negotiation theory and practice [BRE91, RAI82] has been proposed to deal with how people negotiate in real world situations, but we still lack a formal and full understanding on bargaining so that automated processes can be developed. There are some practical negotiation models [OSG62, PRU81] proposed by experimental and social psychologists, but it is difficult to implement their negotiation theories and models in computerized software systems. Current implemented automated negotiation systems for the bargaining type of negotiation are still in a primitive stage [SIE97]. The prototype systems only support single attribute (e.g., price) bargaining and use the approach of hard-coding a small number of simple negotiation strategies which are not effective and efficient in real business bargaining situations. In fact, compared with the popularity of auction in commercial use and research projects, automated bargaining systems are far behind. Although it is relatively easy to implement an automated auction system (especially, the auctioneer server), Guttman and Maes think auction is not well suited for negotiating among applications, e.g., retail markets [GUT98]. The hostile environment and win-lose strategies could cause consumer dissatisfaction and will eventually damage the long-term objectives of merchants who may obtain some short-term gains from the auction. Generally speaking, auction is more applicable to the transactions of products, the supplies of which are limited, such as antiques, famous paintings, hot stocks, etc. Therefore, this dissertation focuses on the issues of automating the bargaining type of

negotiation. By addressing more complex issues, we believe the methodologies and conclusions reached in this work can be applied to other types of negotiations such as auction and bidding.

2.2 Game Theory

Game theory [BIN99, JON80] is the mathematical study of conflicts. Since negotiation is used for resolving conflicts, game theory has been applied to analyze negotiation processes. It focuses on the predication of whether or not an agreement will be reached and if so, the nature of that agreement. Another focus of game theory is the negotiation mechanism design: the definition of protocols that limit the possible tactics or strategies used by players and the mechanism to achieve a so-called Pareto outcome for all negotiation participants [ZLO96]. A Pareto outcome refers to the solution(s) that produces the maximal function values (e.g., sum, product) for each negotiation party's utility score. This is useful since in game theory it is assumed that the negotiation participants are rational and have complete information about the other players. Obviously, in the case of real world negotiation, such assumptions are usually not valid. The lack of any of the assumptions may lead to a failure predication in the game theory paradigm. [MYE83] has shown that in the situation of incomplete information, a rational game player (negotiator) may fail to reach an agreement despite the existence of an agreement zone. Moreover, since game theory is mainly used to predict the outcome of the negotiation process, it can not determine how to obtain the outcome through interactions among players or how to select an optimal negotiation strategy in a

negotiation process. These observations suggest that game theory will be only of limited use in developing automated negotiation systems.

2.3 Negotiation Support Systems

Negotiation Support Systems (NSS) [LIM93, RAN97] and Group Decision Support Systems (GDSS) [KAR97] represent an extension of Decision Support Systems (DSS) to the area of negotiation. The challenges of negotiation and the shortcomings of human-based negotiation have prompted researchers to pursue computer-supported negotiation, generally known as NSS, which is designed to facilitate the various phases of the negotiation process. Jelassi and Foroughi [JEL89] have developed tools to address behavioral characteristics and cognitive perspectives of negotiators. Woo [WOO90] uses speech act theory to formalize the negotiation process so that machine transmission of messages is possible. A certain degree of automation can be achieved by combining these theories with the appropriate domain knowledge. Benefits could accrue from repetitive, similar negotiations. One example NSS is the concession model of Matwin, Szapiro, and Haigh [MAT91] which hard-wires a general strategy of concession into a multi-issue negotiation support system. Negoplan is another decision support software system developed by the International Institute for Applied System Analysis (IIASA) in Austria [KER98]. It is implemented in Prolog and simulates decision processes by allowing a systematic and analytical solution of sequential decision problems, of which negotiation is an example.

A different NSS by Rangaswamy and Shell [RAN97] employs a computer-based method to elicit a conjoint representation of preferences. Once the parties have a better

understanding of their preferences, they draft proposals electronically based on the support provided by NSS. In controlled experiments, users supported by such an NSS reached better agreements. In addition, the system can suggest a Pareto equilibrium for improved outcomes by observing the offers made by each party and by knowing the preferences of both. This feature raises two issues: the first concerns incentive compatibility and possible strategic behavior. Users who are aware that a computer will make suggestions based on the utility assessment might attempt to manipulate or cheat the system by misrepresenting their preferences. The second concerns trust; users must be comfortable with a central system knowing their preferences and observing their offers.

Although the implementations and the computational approaches are relevant and suggestive, particularly in the areas of system architecture, functional requirements, and user interfaces, an NSS typically emphasizes support, rather than automation.

2.4 Agent Technologies

Distributed Artificial Intelligence (DAI) [OHR96, MUL96] and Multi-Agent Systems (MAS) [ZLO96] are two branches of AI that study coordination, synchronization and interaction among multiple agents. Negotiation is an important way to achieve coordination in DAI and MAS. [ROS94] describes various design conventions for negotiation agents. Kasbah [CHA96, CHA97, KAS99] is a Web-based, multi-agent, classified-ad system where users create buying and selling agents to help exchange goods. These agents represent the buyer, and support and automate much of merchant brokering and negotiation. A user wanting to buy or sell goods can create an agent, initialize it with

a particular negotiation strategy, and send it off into a centralized agent marketplace. Kasbah's agents proactively seek out potential buyers or sellers and negotiate with them on behalf of their owners. Each agent's goal is to complete an acceptable deal, subject to a set of user-specified constraints, such as a desired price, a highest (or lowest) acceptable price, and a date by which to complete the transaction. Negotiation between buying and selling agents in Kasbah is single-issue (price), bilateral, and straightforward. After buying and selling agents are matched up, the only valid action for buying agents is to offer a bid to the seller. Selling agents respond with either a binding "yes" or "no". Given this protocol, Kasbah provides buyers with one of three concessive negotiation strategies: "anxious", "cool-headed", and "frugal" - corresponding to a linear, quadratic, or exponential function, respectively, for increasing a bid for a product over time. The simplicity of these negotiation heuristics makes it intuitive for users to understand how their agents are behaving in the marketplace.

ADEPT [SIE97] is another agent-based automated negotiation system based on the British Telecom (BT)'s business process in which different departments (represented by agents) will negotiate with each other for providing customer quotation requests. ADEPT can support multi-issue, bilateral bargaining type of negotiation. The main functions of ADEPT's negotiation agents include generating an initial offer, evaluating incoming proposals, and generating counterproposals. The intelligence of the negotiation agent includes the evaluation function for each negotiation attribute, a simple aggregation function, and a simple hard-coded monotonic concession negotiation strategy.

In general, negotiation in DAI and MAS usually assumes that agents are cooperative, which may not be valid in real business negotiations. Negotiation agents in

DAI and MAS also lack the necessary human interaction during a negotiation process and are more suitable for applications that require little human interventions: e.g., resource allocation or manufacturing task assignment. Hence, they may not suitable for business situations in which it is hard to totally delegate important negotiation tasks to autonomous agents.

2.5 Machine Learning

The field of machine learning attempts to let software agents learn how to negotiate by themselves, rather than attempting to exhaustively implement human negotiation strategies in software agents. In [Zeng98], Zeng and Sycara present Bazaar, an experimental system for updating negotiation offers between two intelligent agents during a bilateral negotiation. The paper contains a formal analysis of the negotiation state space, which is capable of tracking a rich set of issues and tradeoffs that are necessary for multi-issue negotiation. It explicitly models negotiation as a sequential decision making task and uses Bayesian probability as the underlying learning mechanism. The authors present an example by using price as the issue of the negotiation.

Another learning approach is to use genetic algorithms and genetic programming based on the famous principle of Darwinian evolution. In the context of negotiation, it works as follows: each of the software agents begins with a population of various, randomly generated (and not necessarily good) negotiation strategies. It then employs its strategies against the strategies of the other agents in a round of bargaining, which takes place under specific predetermined rules and payoffs. At the end of a "generation", the agent evaluates the performance of each strategy in its current population and crosses

over strategies from the current "parent" population to create a "child" generation of bargaining strategies. The more successful the strategies have the higher probablitieis they are chosen as parents. Also, mutations may be randomly introduced. The size of the initial population, the number of generations, the crossover rate, and the mutation rate are parameters of the algorithm. In principle, after many trials, the negotiation agent is able to derive a good strategy. However, the actual result depends on many elements, e.g., the number of training trials, the algorithm parameter configuration, and the quality of the negotiation strategy of the training opponent. For instance, to achieve a good strategy, the number of trials varies from about 20 generations [OLI97] to upwards of 4000 generations [DWO95] and all runs must be made against opponent(s) whose strategies are as realistic as possible. Hence, it may be unrealistic to use a genetic algorithm for a real business negotiation because using a human opponent in so many trials may take a long time.

In summary, existing efforts have provided insights into several aspects of automated negotiation, e.g., negotiation protocols, decision support, negotiation theory, and negotiation knowledge acquisition. However, regarding the issues of system architecture, negotiation knowledge representation and execution, interoperability, negotiation process modeling, and negotiation strategy design, comprehensive and effective methodologies are still absent for building a general, scalable, and efficient automated negotiation system to support complex, multi-issue negotiations over the Internet.

2.6 Constraint Satisfaction Problem

Constraint Satisfaction Problems (CSP) have been the subjects of research in Artificial Intelligence for many years. The pioneering works on networks of constraints were motivated mainly by problems arising in the field of image processing [WAL72]. AI research, which has concentrated on difficult combinatorial problems, is dated back to the sixties and seventies and has contributed to considerable progress in constraint-based reasoning. A CSP consists of three main parts: a finite set of *variables*, each of which is associated with a finite *domain*, a set of *constraints* that define the relationships among variables, and the restrictions on the values that the variables can simultaneously have. The task of a CSP engine is to assign a value to each variable while satisfying all of the constraints. Constraints arise naturally in most areas. Sample constraints are: the three angles of a triangle sum to 180 degrees; the position of the scroller in the window scrollbar must reflect the visible part of the underlying document; the meeting must be scheduled at a time period when all participants are available. Thus, constraints are a natural way for people to express problems in many fields. CSP has been successfully applied in numerous domains. Recent applications include computer graphics (to express geometric coherence in the case of scene analysis), natural language processing (construction of efficient parsers), database systems (to ensure and restore consistency of the data), operations research (optimized planning and scheduling of multiple tasks), molecular biology (DNA sequencing), circuit design (component layout) [BAR98], just to name a few. Negotiation can also be formulated as a CSP problem as follows. The *variables* are the negotiation attributes. The *constraints* are the combination of the constraints defined in the received proposal and the pre-registered negotiation

requirements. The task of the CSP engine is to find agreement candidate(s) that can satisfy the requirements of both sides (in bilateral bargaining).

The solution to a CSP can always be found by searching systematically through the possible assignments of values to variables. A more efficient method uses the backtracking [GOL65] that incrementally attempts to extend a partial solution toward a complete solution by repeatedly choosing a value for another variable. Moreover, to overcome the late detection of inconsistencies among constraints while backtracking, various consistency techniques for constraint graphs were introduced to prune the search space. Such techniques range from simple node-consistency, to arc-consistency, to expensive path consistency [FIK68, GAS74]. Pruning is achieved by spending more time at each node of the search tree and removing combinations of values that can not appear in solution.

However, the negotiation problem normally goes beyond the traditional CSP. Because of possible conflicts existing between two negotiators, the CSP engine may not be able to identify an agreement instance that can satisfy all the constraints. These types of problems are called over-constrained problems. Traditional algorithms for constraint satisfaction are not able to solve over-constrained systems. Therefore, some alternative approaches have been proposed to solve over-constrained problems. In the Weighted CSP (WCSP), each constraint is associated with a cost and the goal is to find a solution that can maximize the sum of the satisfied constraint cost. Wallace and Freuder use heuristics to weaken some of the original constraints [WAL96]. However, in the negotiation, it is imperative that the CSP engine can locate the conflicting constraints

precisely so that the violated constraint can be resolved by some conflict resolution strategies. Current research has not successfully addressed this issue.

2.7 Active Database and Rules

Production rules have been used to express the human knowledge in the areas of artificial intelligence and expert systems for a long time. Simple condition-action rules provide a powerful and declarative way to model knowledge in the semantics of "when a condition is true, perform the action". Expert system such as OPS5 [BRO85] and CLIPS [GIA91] use this type of rules. These concepts and rule languages were soon borrowed by the database community to create a new category of databases, namely active databases. Basically, active database systems can be defined as database management systems (DBMS's) equipped with capabilities to process rules or triggers defined on the data. The goal of an active database is to provide declarative rule/trigger definitions and efficient rule processing services for many real database applications. The mainstream models of active database systems correspond to the relational and object-oriented database models. Some examples of these active database systems are HiPAC [DAY88], Ode [GEH91], Sentinel [ChA94], Ariel [HAN96], OSAM*.KBMS [SU93], and Starburst [HAS90]. Ariel focuses on the efficient rule testing mechanism called Gator Networks, an optimized generalization of the original Rete networks. The Starburst rule system processor is based on arbitrary database state transitions, which is in distinction to other rule systems that rely on tuple or statement-level updates. HiPAC was an early, object-oriented, active database project. The project defines three models of event-condition binding: immediate, deferred, and decoupled. The Sentinel project was a follow-on

project to HiPAC and developed mechanisms for 1) monitoring events in a distributed fashion, 2) communication among application using events, and 3) integrating rules into a database programming language. OSAM*KBMS is founded on an object-oriented semantic association model OSAM*, which allows the structural abstractions of any object class to be defined in terms of its various types of associations with other object classes. It also supports the behavioral abstraction in terms of system- and user-defined operations, and knowledge abstraction in terms of knowledge rules with triggers.

The rule language in most of these active database systems adopts the format of Event-Condition-Action (ECA) which provides better control of condition evaluation and rule invocation in the context of database operations, e.g., delete, insert, and update. Some variations of ECA rules further improve the expressive capability and flexibility of rule languages. Event-Condition-Action-Alternative Action (ECAA) [LAM98] adds alternative actions to be taken when the condition is evaluated to be false. In our work on automated negotiation, we separate events from condition-action-alternative-action rules and use triggers to relate events with rules to provide the flexibility of linking different events with different rules and rule structures. We use events, triggers and rules to specify the negotiation strategies of human negotiation experts. We integrate the rule definition and processing facility with other components of the automated negotiation system, including the constraint checking.

2.8 XML and BOD

To enable effective communication among replicated negotiation systems, the negotiation messages that are exchanged between the systems must adhere to a standard

format. In this area, two important initiatives should be mentioned. The first important standard is the eXtensible Markup Language (XML) established by the World Wide Web Consortium in 1998 [W3C98]. It is a data format for semi-structured document. As opposed to HTML, which is a specific markup language for visual presentation of a document, XML has the ability to express both syntax and semantics of the data. When both semantics and visual presentation of the data are needed, the World Wide Web Consortium provides a solution in the form of an eXtensible Stylesheet Language (XSL). XSL is the bridge between XML and HTML that specifies how constructs in XML documents should be displayed on the screen.

The power of XML lies in its extensibility. People are able to define a set of domain-specific tags that carry the semantics of the documents. The most important part of XML-based application development is the definition of Documents Type Definition (DTD) files for the application domain. A DTD file is essentially a context-free grammar in the Extended BNF (Backus Naur Form). A DTD allows the specification of optional/required elements, multiple occurrences (zero or many and one or many) of elements and default values. XML describes how the tags are structured into "well-formed" or "valid" XML documents.

The second standard is the Business Object Document (BOD) stanadard specified by the Open Application Group (OAG) [OAG99]. OAG is a consortium to promote the easy and cost-effective integration of key business application software components for enterprise and supply chain management systems. The BOD is used to communicate a request and data from a source business application to a target business application(s). Each BOD includes supporting details to enable the destination business application(s) to

accomplish the action. Structurally, a BOD consists of two parts: a control area and one or more business data areaS (BDA). Figure 2-1 shows the BOD structure.

The business data area contains all the codes, parameters and values needed to support the business service request. For example, to post an inventory receipt, the business data area contains all the accounting information needed for the G/L (General Ledger) application. A BOD may contain one or many BDAs supporting the same BSR (Business Service Request). For example, a business service request to post general ledger journal entries may contain one or many journal entries, each transferred in one BDA of the same BOD.

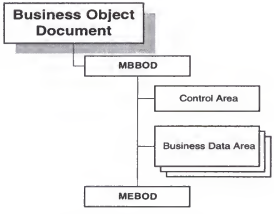

Figure 2-1: BOD Structure

OAG has released hundreds of standard business object documents to enable inter-enterprise operations. However, none of them is designed for negotiation, although negotiation is an important part of business activity. In this dissertation, we utilize both standards (XML and BOD) to enable the interoperability between heterogeneous negotiation systems. Interoperation is achieved by defining a set of XML-based

negotiation Business Object Documents, which are currently not defined by OAG. Note, we are, however, attempting to propose to OAG that our negotiation BODs become part of the standard.

CHAPTER 3
WEB-BASED AUTOMATED NEGOTIATION SERVER OVERVIEW

In this chapter, we present an overview of the web-based automated negotiation server. Section 3.1 presents the system architecture and Section 3.2 outlines the build-time and run-time automated negotiation process.

3.1 System Architecture

In order to take full advantage of the Internet as a public resource and infrastructure for E-Commerce which can be easily accessed by a large number of users, we propose a web-based automated negotiation system (WANS) architecture which is a general, open, and interoperable automated negotiation framework. The architecture design is depicted in Figure 3-1. To simplify our presentation of the negotiation process in the later sections, we use the symbols EA, UA, and NSA to denote the expert, user, and negotiation server of the negotiation party A, respectively. The corresponding symbols EB, UB, and NSB are used to denote those of negotiation party B. EA and EB refer to those users who define negotiation policies and strategies from business perspectives. Corporation executives or negotiation experts are likely to perform this role. UA and UB refer to those users who start the negotiation process after the negotiation policies and strategies have been input into the negotiation system, monitor the automated negotiation process, override the actions suggested by the negotiation server, and provide the server additional information needed for it to continue its process.

In addition, some clients may have application systems that maintain the inventories of goods or information regarding the services they provide.

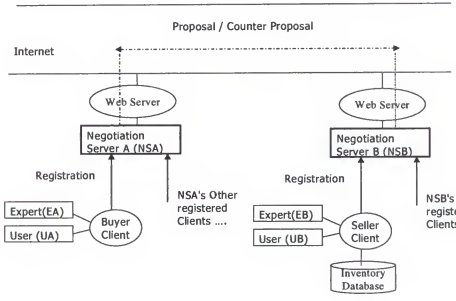

Figure 3-1: Web-Based Automated Negotiation System Architecture

Similar to Web servers, which provide many useful services, each negotiation server provides the following fundamental negotiation services:

- A *registration service* to allow a buyer to specify the requirements and constraints in terms of the goods or services he wants to acquire, and a seller to specify the information and constraints related to the goods and services he provides. A client can also register a set of negotiation rules, which specify the negotiation strategies/tactics to be followed when constraints are violated during the negotiation phase. Additionally, he can specify the preference scoring and aggregation methods to be used for the cost-benefit analysis of alternative combinations of negotiation

conditions. The specification of a product or service and the specification of a negotiation proposal or counter-proposal are posed in a content specification language whose underlying object model is an active object model (AOM) [LEE99], which represents all objects of interest in terms of attributes, operations (methods), events, rules, and triggers. The rule specification part of the language is an extension of the Event-Condition-Action (ECA) rule paradigm [WID96]. The registered information is stored persistently by the negotiation sever.

- A *negotiation proposal processing service* to evaluate proposals and generate counter-proposals, explanations, messages, etc. by following a negotiation protocol. Negotiation continues until an agreement is reached or the negotiation is terminated unilaterally by a client. A constraint satisfaction processing component is developed to evaluate a negotiation proposal or counter-proposal against pre-registered requirements and constraints. An event is posted to signal the detection of a constraint violation.
- An *event and rule management service* to detect and manage events and to trigger rules that are relevant to the posted events. These rules may automatically relax some constraints, call for human interventions or perform other operations. The relaxed constraints and/or human inputs are used as the basis to form counter-proposals. This service is provided by an Event-Trigger-Rule (ETR) server [LAM98]. Contrary to some existing approaches in which negotiation strategies are hard-coded in the form of agent programs, we use a high-level rule specification language and GUI tools to allow human negotiation experts to dynamically add and change negotiation rules at run-time.

- A *cost-benefit analysis service* to allow a client to perform quantitative analysis on alternative negotiation proposals and to obtain their relative, quantitative cost-benefit measures. For this service, we have adopted the cost-benefit decision model presented in [SU87].

The negotiation server can be replicated and installed on multiple Web sites to provide automated negotiation services for multiple users and/or business organizations on the Internet. Buyers and sellers (henceforth referred to as clients) can select their own trusted negotiation servers to conduct negotiations on their behalf (the same way as lawyers are asked to represent their clients in legal matters). The architecture is general to support other types of negotiations, e.g., auction, bidding although functionality of our negotiation server is illustrated in the context of bilateral bargaining. For instance, in bilateral bargaining, two negotiation servers will act as bargaining parties and negotiate with each other. In auction, one server will be the auctioneer and other servers represent the bidding clients. Furthermore, the architecture is an open framework on the Internet. The negotiation server can interact with other negotiation systems as long as both of them can conform to the interoperable interface.

3.2 Automated Negotiation Process Overview

Conceptually, we envision the negotiation system to work as follows (shown in the Figure 3-2).

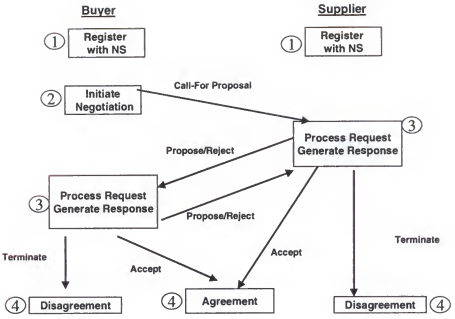

Figure 3-2: Bilateral Bargaining Process

Before an automated negotiation process occurs, a client must inform a selected negotiation server and, indirectly, the other parties involved, of the goods and services he provides (in the case of a supplier) or wishes to acquire (in the case of a consumer) as well as any special requirements that may influence the negotiation (e.g., delivery constraints or special discounts for large orders). For sellers, part of this registration procedure is accomplished through advertisements which are published as web pages. The contents of these advertisements can be indexed and catalogued, for example, by application domains and are searchable just like regular Web pages. We assume that negotiation servers either maintain their own index of relevant advertisements or use a third-party tool, such as a search engine (e.g., AltaVista [ALT99]), yellow page services (e.g., Yahoo [YAH99]), or trading services to help locate the advertisements ("the

information pull model"). Alternatively, one can also envision a scenario in which sellers actively inform the relevant negotiation servers of their goods and services by sending the advertisements directly to the servers ("the information push model"). The approach described in this dissertation is independent of the particular paradigm (i.e., either push or pull) used by sellers. The buyers also need to submit a set of requirements and constraints describing the desired goods or services to a negotiation server of their choice. In addition, at build-time, users register necessary negotiation knowledge and intelligence, e.g., negotiation strategies and cost-benefit models, with web-based negotiation servers that can be used for the decision-making in the negotiation.

At run-time, negotiation servers represent their registered users and make use of the knowledge to conduct automated negotiations. The run-time automated negotiation process can be explained by using bilateral bargaining as an example. A buyer client (i.e., UA in Figure 3-1) initiates the negotiation process by submitting a call-for-proposal (CFP) to the selected negotiation server. The negotiation server (NSA) carries out the negotiation process by sending the CFP to NSB, which represents the seller UB. NSB checks the conditions specified in the CFP against the registered requirements and constraints of UB. If they are consistent (i.e., no violation of constraints), NSB will send an acceptance (before the actual acceptance human involvement may be needed depending on the authority configuration). In the event that a conflict is found, relevant strategic rule(s), which have been provided by a negotiation expert (EB), are applied to take following alternative actions:

(1) Reject the proposal and indicate the violated conditions in the received proposal.

(2) Make changes to the conditions and constraints of the received proposal and send the modified proposal back to NSA as a counterproposal.

(3) Consult with UB for decision-making.

In the event UB maintains an inventory of goods (e.g., a database system with a public interface), NSB can use the data and constraints specified in the proposal to query the database to verify the availability of the goods and services in question. Alternatively, it may use a more complex planning and scheduling system to answer questions about the availability or to provide the data requested in the received proposal. Should multiple alternatives be generated, a cost-benefit analysis component is called upon to perform quantitative analysis and to provide a cost-benefit rating of the alternatives. After the evaluation, the highest ranked alternative will be sent back to NSA as the counter-proposal. This may trigger another round of negotiation. This process will continue until an agreement is reached or either side terminates the negotiation process.

CHAPTER 4
NEGOTIATION KNOWLEDGE ACQUISITION AND REPRESENTATION

This Chapter presents a systematic, efficient and declarative negotiation knowledge acquisition and representation mechanism provided by the automated negotiation server. Through a set of graphic user interfaces (GUIs), human negotiation experts can register necessary negotiation knowledge, including negotiation requirements, strategies, and cost-benefit evaluation models, with the negotiation system. The knowledge specification language is declarative and easy to use. The internal knowledge representation adopts the object-oriented model to enable efficient processing. All of the registered knowledge can be made persistent in an object store. At run time, the negotiation system can then utilize the knowledge to automatically conduct effective negotiations and provide necessary negotiation decision support to the human user. In the following sections, we illustrate our approaches to negotiation knowledge acquisition and representation in detail. Specifically, section 4.1 describes the object-oriented constraint-based requirement specification language. Section 4.2 presents the Event-Trigger-Rule-based negotiation strategy specification. Section 4.3 discusses the cost-benefit analysis and evaluation model.

4.1 Object-Oriented Constraint-Based Requirement Specification

As stated previously, before the negotiation can begin, the user must provide the negotiation server with necessary negotiation requirements. Negotiation requirements

refer to the business constraints and policies that need to be satisfied in the negotiation. It is used to evaluate the incoming proposal to check whether the proposal satisfies the constraints and policies of the client which it is representing. Based on the result of the evaluation, a decision of rejection or acceptance can be made. Furthermore, the negotiation requirements are the basis for the proposal content since any proposed proposal must reflect the client's business policies and constraints also.

In order for an arbitrary negotiation server to be able to communicate with its clients and process their registration information, the requirement specification must be formulated in a common language (requirement 1) using a vocabulary that is understood by all parties involved (requirement 2). In order to satisfy the first requirement, we have developed a content specification language and its accompanying Web-based GUI tools for expressing the registration information, i.e., attributes, data values, and constraints. In order to satisfy requirement 2, we use an ontology [FOX94] for describing the terms and their relationships that are used during the registration as well as those used in the actual negotiation process. In this dissertation, we assume that a common ontology exists and is used by negotiation participants, i.e., sellers, buyers, and negotiation servers.

In general, there are two options for specifying the negotiation requirements, namely, constant-value based and constraint-based. In the first option, for instance, when buying a PC, the negotiator needs to provide constant values for the PC's configuration information and business contract attributes, e.g., brand name, CPU type, memory size, delivery date, unit price, etc. Obviously, this approach lacks flexibility for the following reasons. First, a client may not have the domain knowledge to give a value for each attribute. For example, the user may not have necessary knowledge to specify the latest

CPU type available on the market. Second, a client may not have enough knowledge to specify a combination of multiple attribute values. For instance, determining the proper CPU - memory configuration might be a difficult task for many customers. In this case, it is more convenient for the clients to specify a range of values or enumerate alternatives for these attributes. Finally, a fixed value for some attributes may not reflect a client's needs. For instance, a fixed delivery date may be unimportant to a consumer. Instead, he or she can accept a range of delivery days by giving a lower bound and an upper bound for the delivery day requirement.

To overcome the shortcomings of the constant value based requirement specification, in this dissertation, we introduce a constraint-based, object-oriented content specification language (OOCSL). In OOCSL, we first define a negotiation domain using an object-oriented model in which each product or service offered (supplier) or to be acquired (buyer) in the negotiation is represented as an object entity containing one or more attributes describing its properties. The negotiation requirement with respect to the product or service is defined as a set of constraints, including attribute constraints and inter-attribute constraints, e.g., "quantity less than 20", "if quantity is less than 20, the price must be greater than $10". Obviously, this type of semantics can not be expressed in a requirement specification that allows only constant values.

The following example shows the requirement specification of a computer system entity provided by a supplier.

Computer System Supplier's negotiation requirement specification

```
ENTITY Computer_System {

ATTRIBUTE_CONSTRAINT:
      model         String  ANY                                priority [3]
      memory        Integer ENUMERATION[32m,64m, 96m]          priority [4]
      monitor       Integer EQUAL 17                           priority [5]
      hard_drive    Integer ENUMERATION[4g, 6g, 8g]            priority [6]
      unit_price    Float   DERIVED                            priority [2]
      deliver_day   Integer RANGE[14..21]                      priority [1]
      quantity      Integer RANGE[250..550]                    NONNEGOTIABLE

INTER_ATTRIBUTE_CONSTRAINT:
      iac1   quantity >= 400 implies deliver_day >= 16         priority [1]
      iac2   memory < 64m implies hard_dirve < 4g              priority [1]
}
```

In this example, the supplier is advertising a computer system he can provide. The computer system is available in many different configurations as described by the attributes listed underneath the keyword **ATTRIBUTE_CONSTRAINTS**. Each attribute value is of a particular (atomic) type (e.g., **String**, **Integer**). An attribute constraint is specified by **ENUMERATION** a set of possible values or by a value **RANGE**. If the attribute has only one possible value, the keyword **EQUAL** is used. In addition, attributes whose values can not be determined until the negotiation is under way are marked as **DERIVED**. Some attribute values are marked as **ANY**, which means "ask for the value during negotiation" or "do not care". All attribute values are negotiable unless marked by the special keyword **NONNEGOTIABLE**. The **priority** values rank the importance of different constraints and are used to decide the order in which constraints are checked. **NONNEGOTIABLE** attribute constraints are always verified first.

Inter-attribute constraints describe the relationships between two or more attributes. The syntax for inter-attribute constraints is:

Constraint-name Antecedence ***implies*** *Consequenc*

In the example above, the inter-attribute constraint called `iac2` specifies that if a customer wants to buy a computer system with memory size less that 64 megabytes, the corresponding hard drive capacity must be under 4 gigabytes.

The complete BNF of OOCSL is given in the Appendix A. Similarly, the buyer can submit the purchase requirements to his selected negotiation server as follows:

```
ENTITY Computer_System {

ATTRIBUTE_CONSTRAINT:
      model         String      EQUAL "PentiumII 350"      priority [7]
      monitor       Integer     ENUMERATION[15, 17, 19]    priority [6]
      memory        Integer     RANGE[16m..64m]            priority [5]
      hard_drive    Integer     RANGE[1g.. 12g]            priority [4]
      unit_price    Float       EQUAL 1700.00              priority [3]
      deliver_day   Integer     RANGE[3..10]               priority [1]
      quantity      Integer     RANGE[300..400]            priority [2]

INTER-ATTRIBUTE-CONSTRAIN:
      Constraint1  monitor = 15 implies unit_price < 1500  priority [1]
}
```

The end user actually does not need to provide an OOCSL script for his requirement specification. Instead, a web-based graphic user interface (GUI) is provided to guide the user to input the negotiation requirement constraints. A snapshot of the requirement registration GUI is shown in Figure 4-1, which describes another set of negotiation requirements provided by a seller of EtherNetCards. The requirements include both attribute and inter-attribute constraints. For example, we can see that the seller would like to sell between 250 and 550 cards (quantity constraint) within 2 to 3 weeks after the closing of the deal (delivery_period constraint). The sole inter-attribute constraint specifies that if the quantity reaches 400 cards, then the delivery_period must be more than 16 days after the closing date of the deal.

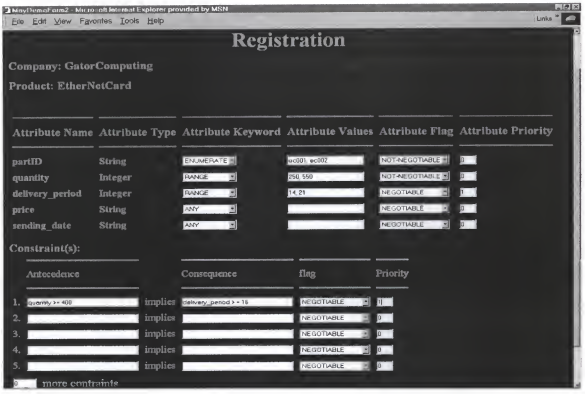

Figure 4-1: Snapshot of the Web-Based Requirement Registration Interface.

The above negotiation requirement registration GUI is extensible to support multiple negotiation items. In fact, it is a dynamic HTML web page which is generated based on the ontology information of the negotiation items. If the ontology is changed or the user needs to register negotiation constraints for other negotiation items, the user just needs to select the item name from the RegistrationFormGenerator GUI (shown in Figure 4-2) and the corresponding registration form for the selected negotiation item will be presented.

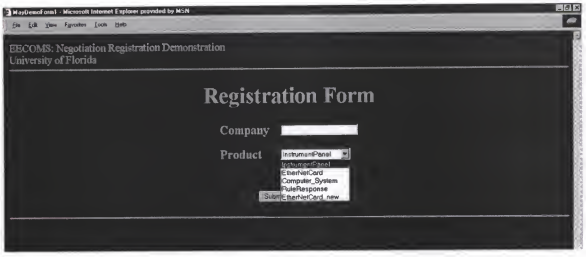

Figure 4-2: Product Selection in the Registration

The OOCSL and GUIs facilitate the negotiation requirement registration process. Furthermore, to improve the efficiency for the subsequent negotiation process (e.g., constraint consistency checking), we defined an object representation for the requirement constraints. Figure 4-3 shows the object graph of the class Constraint used in our negotiation server implementation. To simplify the presentation, the operations associated with each object are not shown in the diagram. As we can see from the object graph, corresponding to the OOCSL, the Constraint class consists of three attributes, namely product (the negotiation item for which the constraints are defined), interAttributeConstraintList, and attributeConstraintList. Both lists include a set of inter-attribute and attribute constraints respectively. The AttributeConstraint class contains the necessary attribute constraint information defined in the OOCSL, e.g., attribute name, data types, priority, keyword, etc. Similarly, the InterAttributeConstraint class contains the inter-attribute constraint information, antecedence, consequence, etc. By making use

of the object-oriented data model, the constraint information is easier to be retrieved, modified, and processes by the negotiation server.

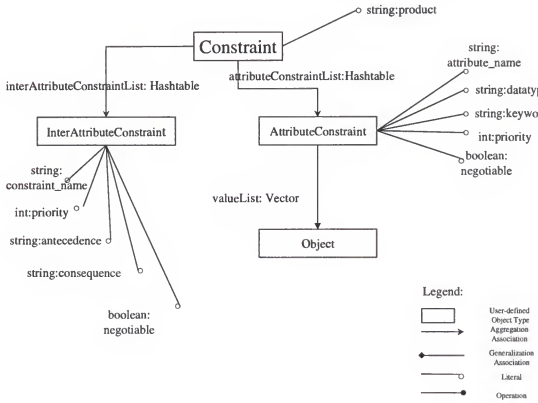

Figure 4-3: Object Graph of the Constraint

4.2 Rule-based Negotiation Strategy Specification

The second important part of the registration procedure is the specification of negotiation strategies or tactics to be used by the negotiation server on behalf of its clients. Unlike most existing work which is based on hard-coded negotiation strategies, in our negotiation system, we adopt an Event-Trigger-Rule (ETR) based approach to specify and represent the human-based negotiation strategies in a declarative fashion.

As previously stated, negotiation strategies are mainly used to resolve conflicts (e.g., violated constraint). We define each conflict as an event which can be associated with each attribute or inter-attribute constraint. The name of the event is generated by concatenating the name of the user who defines the constraint, the class names of the negotiation items, and the attribute name or inter-attribute constraint name.

Each strategy is expressed in terms of a rule, which specifies the condition to be checked and the actions to be taken by the negotiation server. Rules are activated upon the occurrence of specific events (upon detecting a conflict) during the negotiation process. A strategic rule has three parts: (1) the condition under which the subsequent action is to be taken (e.g., the proposed sales price is below manufacturing cost); (2) the particular action(s) to be taken when the condition is met (e.g., the modification of the current constraint); (3) an optional alternative action(s) to be taken when the condition is not met (e.g., a request for human intervention).

The relationship between events and rules is specified by triggers. A trigger specifies that, upon the occurrence of any number of alternative events (called trigger events), a composite event or event history should be checked and a structure of rules may be activated if the composite event verification is successful. Events and rules are referenced by their names.

At runtime, when a violation of the attribute or inter-attribute constraint is detected, the corresponding event is posted to trigger the corresponding strategic rules to resolve the conflicts. Compared with the traditional logic and rule specification language, e.g., Event-Condition-Action (or ECA) rules, the separation of events, rules and triggers in the ETR paradigm provides more flexibility in linking trigger events and event histories with

rules or rule structures. For example, a rule name may appear in several rule structures, which are triggered by the occurrences of different events. In a traditional ECA system, such a rule would have to be specified repeatedly using multiple ECA rules. Furthermore, the ETR specification makes a more explicit semantic distinction between trigger events, the verification of composite events, event history, and the rule structure than traditional ECA rules. To our knowledge, all current ECA systems treat the events of a composite event as trigger events, i.e., the occurrence of any one of these events will trigger the evaluation of the composite event. However, in some situations, only some of these events should trigger the evaluation of the composite event and its associated rules.

The following four strategic rules outline the specification of negotiation strategies in the ETR format. We refer to the entity class Computer_System in the sample advertisement given in Section 4.1 as the context. Since the sample negotiation rules are very simple (no composite events), instead of using the full syntax of ETR rule specification [LEE99], we only show the trigger events and rules that constitute trigger specifications.

Supplier Strategic Rule 1: *If* `deliver_day` is out of range, check the lower bound of the constraint for attribute deliver_day. If the lower bound is still greater than 10 (i.e., the bottom line), shorten the delivery time by two days. Otherwise, the constraint is considered unResolvable and human intervention is required.

```
TriggerEvent: Supplier%Computer_System%delivery_day
SR1:          Condition: getLowBound("delivery_day") > 10;
              Action:    downLowBound("delivery_day", 2);
              Alternative: unResolvable("delivery_day can not be satisfied");
```

Supplier Strategic Rule 2: If the *iac2* constraint is violated, check if this constraint has already been relaxed. If not, relax the constraint to: "quantity >= 500 **implies** delivery_period>=15" and change the relaxation flag to true to prevent a further relaxation. Otherwise, the constraint is considered unResolvable and human intervention is required.

```
TriggerEvent: Supplier%Computer_System%iac2
SR2:          Condition: getIACStatus("iac2") = "NOCHANGE";
              Action: setIAC("iac2", "quantity >= 500",
                             "delivery_period>=15");
                      setIACStatus("iac2", "RELAXED");
              Alternative: unResolvable("delivery_period or quantity can not be
satisfied");
```

Buyer's Strategic Rule 1: *If* `quantity is` *out of range, check the lower bound of my constraint for* `quantity`. If that value is greater than 250 which is my bottom line, I am willing to reduce the quantity by 100. Otherwise, the case is unResolvable andhuman intervention is required.

```
TriggerEvent  Buyer%Computer_System%quantity
BR1:          Condition: getLowBound("quantity") > 250;
              Action:  downLowBound("quantity", 100);
              Alternative: unResolvable("quantity can not be satisfied");
```

Buyer's Strategic Rule 2: *If* `deliver_day` *is out of range,* check if this constraint was previously relaxed. If not, I am willing to extend the delivery_day by two days. Otherwise, the case is unResolvable and human intervention is required.

```
TriggerEvent  Buyer%Computer_System%delivery_day
BR2:          Condition: getACStatus("delivery_day") =" NOCHANGE";
              Action: upHighBound("delivery_day", 2);
                      setACStatus("delivery_day", "RELAXED");
              Alternative: unResolvable("delivery_day can not be satisfied");
```

ETR-based strategic rule specification provides a rich set of semantic capabilities for negotiation experts to express their negotiation strategies. As we have shown in the

above example, four typical strategies have been expressed. Rule SR1 and BR1 are gradual relaxation strategies and rule SR2 and BR2 express one-time relaxation strategies. We also can observe that both sides use a combination of cooperative and competitive negotiation strategies. "Cooperative" means that the negotiator is willing to relax his constraints when constraint violations are detected. "Competitive" means that the negotiation strategies try to protect the negotiator's business interest at the bottom line value. If the bottom-line value is reached and the violation still exists, a rejection message will be sent out including the reason for the rejection. In this case, it is the other side's turn to decide if a concession should be made.

Other types of negotiation strategies and methods of generating counterproposals can also be specified in ETR rules. For instance,

(1) Trading. Exchange one preference for another. For example, if the the two sides disagree over the price, the price constraint will be relaxed. However, to protect the overall interest, another negotiation attribute, say, delivery schedule may be adjusted to the counterpart's favor. This can be specified in ETR rules because multiple action statements are allowed.

(2) Packaging. One straightforward way of evaluating constraints is to do it one by one and to trigger strategic rules when a conflict is detected. However, there can be dependencies among constraints. Sometimes, it may be a better approach to evaluate a number of related constraints and apply their corresponding rules (i.e., packaging the constraints) to relax the constraints together to form a counterproposal. In ETR paradigm, although an event only represents one constraint evaluation status, namely violated or consistent, a trigger specification

can link a single negotiation strategy rule or rule structure with multiple events. The relationship between events can be further expressed by event operators, such as logic operators (AND and OR), and time or sequence operators (BEFORE, AFTER, etc.). Thus, the packaging of constraint evaluation and relaxation can be handled.

(3) Appeal to authority. Negotiation is a complex decision making process. Sometimes, computational intelligence is too limited to resolve certain conflicts. Therefore, human involvement is required in some situations. However, constant human involvement is tedious and will slow down the negotiation system performance. Strategic rules allow the insertion of the proper human intervention points into the ACTION part of ETR rules based on the predefined conditions specified in the CONDITION part.

To assist the human negotiation experts to specify the negotiation strategies, including defining the events, triggers, and rules, we also provide a set of web-based GUIs. Figure 4-4 shows a rule definition GUI

In summary, ETR-based strategic rule specification provides a declarative way for negotiation experts to express their negotiation strategies or tactics that will be used by the automated negotiation system to resolve the conflict, and to control the process of the negotiation at runtime. Compared to a hard-coded program, the ETR based negotiation strategy specification is more expressive, easier to maintain, and easier to change.

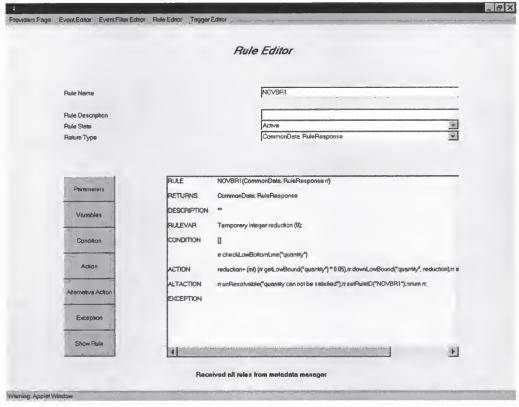

Figure 4-4: Web-based ETR Rule Specification GUI

4.3 Cost Benefit Analysis Model for Proposal Evaluation

Negotiation is a complex decision-making process. The decisions may not only result in the rejection or acceptance of a proposal, but also affect the evaluation of and the selection from multiple choices. For example:

- A client may receive a proposal, which specifies a number of alternative negotiation conditions.
- A client may conduct simultaneous negotiations with multiple parties.

- A single proposal may contain a large number of value combinations which need to be evaluated to make the selection (since a proposal uses range, enumeration, and constraints to specify purchase or sales requirements).

In this case, automated negotiation systems need to decide the trade-off between different alternatives. Sometimes, it is relatively easy to make a selection. For example, if the main concern for the buyer is the price, he can just compare the alternatives and select the one having the lowest price as the deal. However, when multiple negotiation attributes are considered, the selection process may become complicated. For example, a buyer of a computer system receives three proposals from three different suppliers containing two negotiation attributes: price and delivery schedule.

- Supplier A: (price: $1950, delivery: 14 day)
- Supplier B: (price: $1850, delivery: 7 day)
- Supplier C: (price: $1700, delivery: 16 day)

Since the three proposals have different advantages and disadvantages with respect to the two negotiation attributes, it is difficult to judge the trade off between alternatives and to make a correct decision without the support from a systematic and quantitative evaluation tool that can produce a single quantitative evaluation score.

In this dissertation, we have adapted the cost-benefit analysis and decision model (CBADM) reported in [SU87]. In this model, the contents of each proposal instance are divided into two structures: (1) the cost structure, which contains all the attributes to which costs can be assigned (e.g., a specific disk, an additional memory board, etc) and (2), the preference structure, which contains all the attributes to which preference scores can be assigned subjectively (note: these two sets of attributes may overlap). The

structures are separately analyzed to obtain two aggregated values: an aggregated cost value and the global preference score. These two values are then combined to derive a global cost-preference indicator for each combination of data conditions in a proposal instance. The preference scoring and aggregation functions associated with all the attributes are specified by clients during the registration process. Costs associated with different values of these attributes can usually be found in an inventory system.

In the preference analysis based on the preference structure, preference scores ranging from zero to one are assigned to all the attribute-value pairs using a set of "elementary criteria." An elementary criterion is a mapping from an attribute-value pair to a real number between zero and one. The real value expresses a client's degree of satisfaction for the particular attribute value. For example, if Customer-Service is an attribute in the negotiation of computer system purchasing, a client may assign a preference score of 0 if the value of Customer-Service is "mornings only", 0.3 if it is "daytime-only", and 1 if there is 24-hour service. Here, the score of 1 means one hundred percent satisfaction. For attributes whose values can not be enumerated, e.g., Mean-time-to-failure of a disk, mathematical evaluation functions can be defined and used as the elementary criteria. The elementary preference scores are then weighted and aggregated into a global preference rating using a spectrum of "preference aggregation functions", which are derived from a weighted power mean [DUJ75]:

$$E=(w_1 .e_1^r + w_2 .e_2^r + ... + w_n .e_n^r)^{1/r}$$

By varying the value of *r*, a spectrum of aggregation functions is generated, including functions such as *min*, *max*, *weighted arithmetic mean*, etc. Some commonly used functions are given in Table 4-1 below.

Table 4-1 Aggregate Function Spectrum.

Minimum	E= min(e1, e2, … ,en)	r = -∞
Harmonic mean	E=1 / (w1/e1+w2/e2+… + wn/en)	r = -1
Geometric mean	$E=(e_1)^{w1}.(e_2)^{w2}$....	r=0
Weighted arithmetic mean	E=w1*e1 + w2*e2 +… wn*en	r=1
Square mean	$E=\sqrt{w_1*e_1^2+w_2*e_2^2+...}$	r=2
Maximum	E= max(e1, e2, …., en)	r=+∞

These alternative aggregation functions represent different degrees of conjunction and disjunction of negotiation data conditions. They can be selected by clients to suit different decision situations and to aid in the selection of different products and services. For example, the maximum aggregation function is suitable when one or more of the negotiation conditions are acceptable to a client. In that case, the maximal value among all the preference scores derived for the negotiation conditions will be used as the global preference score. For example, if CPU speed is most important to a client and he is 90% satisfied (i.e., preference score of 0.9) with the speed of the computer under consideration, the preference scores of the rest of attributes can be ignored. In this case, the global preference score is 0.9. On the other end of the spectrum, the Minimum function would use the minimal score among all the preference scores as the global preference. In the above example, if a client is only 10% satisfied with the speed of the CPU, the global score is 0.1 even though he may be totally satisfied with all other attribute values. For simplicity, we also can use the integer value in the range of 0 to 100 to represent the percentage of elementary score and global score.

During the negotiation, if the customer receives multiple alternatives, the customer can then apply the model to analyze different alternatives to make a selection based on the

evaluation results. The following is a sample cost-benefit model specified by a presumed buyer in a computer system purchasing negotiation. The model includes two elementary scoring functions for two negotiation attributes, price and delivery schedule and an aggregation function of arithmetical mean.

Table 4-2 Elementary Scoring Function on Price

Proposal price	Score
>=2000	0
>=1900 AND <2000	30
>=1800 AND < 1900	70
<1800	100

Table 4-3 Elementary Scoring Function on the Delivery Day

Delivery_day	Score
>14	20
>10 AND <=14	60
>7 AND <=10	80
<=7	100

The aggregation function is as follows:

```
Gps = Ps (proposal.price)* 0.6 + Ps(proposal.deliver_day)*0.4
```

Based on the definitions of the elementary scoring function, we can see the preference of the customer is as follows: low price and fast delivery. Based on the aggregation function, we can see that the customer ranks the preferences of the price attribute higher than that of the delivery schedule attribute. Both are reasonable considering the value-oriented nature of most shoppers. Regarding the problem described

at the beginning of this section, after the buyer apply the model to make a selection, the evaluation result for the three proposals from three suppliers will be:

- Supplier A: (price: $1950, delivery: 14 day) Score: 42
- Supplier B: (price: $1850, delivery: 7 day) Score: 82
- Supplier C: (price: $1700, delivery: 16 day) Score: 68

Based on the result, the customer should choose B as his preferred agreement because it gives the best score according the configured cost-benefit analysis model.

Again, we provide a set of web-based GUIs to assist the user to configure his cost-benefit model for different negotiation items, including specifying the elementary functions for negotiation attributes and the aggregation functions shown in Figure 4-5.

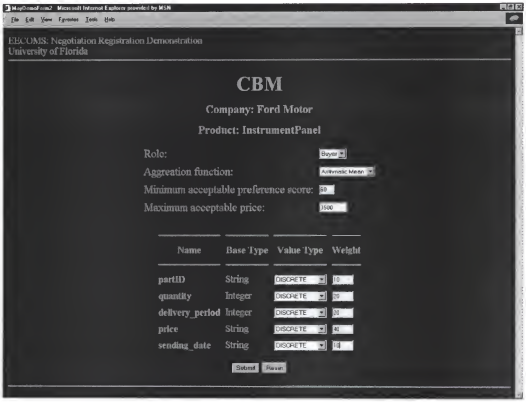

Figure 4-5: Cost-Benefit Evaluation User Interface

CHAPTER 5
AUTOMATED NEGOTIATION PROCESS

Based on the negotiation knowledge captured from users during build-time and modified during run-time, a negotiation server can then utilize the knowledge to conduct negotiations automatically on behalf of the users, including communicating with the counterpart negotiation server, evaluating incoming proposals, and generating counter-proposals. Section 5.1 presents a set of negotiation primitives and a formal bargaining protocol to enable effective communications between negotiation servers. Section 5.2 presents the general structure of negotiation messages expressed in XML BODs. Section 5.3 describes the negotiation decision process, including evaluating a proposal, resolving conflicts, making concession, and analyzing and selecting the best alternative based on cost-benefit evaluation scores. Section 5.4 presents the algorithm used by the constraint satisfaction processing engine to support automatic proposal evaluation. Section 5.5 introduces the technique used by the Event-Trigger-Rule (ETR) server to support the dynamic execution of rules which implement negotiation strategies. Section 5.6 gives some sample negotiation process scenarios.

5.1 Negotiation Primitives and Negotiation Protocol

5.1.1 Negotiation Primitives

To automate the negotiation process, it is important to ensure that the negotiation servers can communicate automatically and effectively. To enable effective communication between the negotiation servers, we need to define a set of negotiation primitive operations, which are meaningful message types that can be exchanged between the servers during a bargaining process. There are some existing efforts in defining such set of primitives. Muller has given a list of negotiation primitives used in the DAI applications [MUL96]. The Agent Communication Language (ACL) [FIP 96] and the Knowledge Query and Manipulation Language (KQML) also define some primitives that are useful for agent negotiations. However, many primitives are for general agent communications and are not germane to negotiation. The set of negotiation primitives is not complete and their semantics are not sufficient to conduct the bargaining-type negotiations in E-Commerce since they are primarily designed for auction and bidding protocols. In this work, we extend the existing set of negotiation primitives and, in some cases, redefine their semantics to support the bargaining-type negotiation. The extended set is shown below in Table 5-1.

Table 5-2 Negotiation Primitives

PRIMITIVE NAME	SEMANTICS DESCRIPTION
CFP	Initiate a call-for-proposal
Propose	Send a proposal or a counter-proposal that specifies the constraints and conditions acceptable to the sender.
Accept	Accept the terms and conditions specified in a proposal without further modifications.
Terminate	Unilaterally terminate the negotiation process.
Reject	Reject the received proposal with or without an attached explanation and expect the counterpart to modify and resent the proposal
Acknowledge	Indicate the receiving of a message
Modify	Modification to the proposal sent last
Withdraw	Withdraw the latest proposal

5.1.2 Negotiation Protocol and State Transition Diagram

To ensure effective and meaningful communication between negotiation servers, these servers must follow a well-defined protocol to exchange messages in a bargaining process. For instance, after the buyer server sends out a CFP, it expects to receive a Propose from the counterpart. Other types of messages would be invalid. As we have discussed in Chapter 2, several negotiation protocols for bidding and different types of auctions are available and have been used in some automated systems. In this work, we use two state transition diagrams to define a bilateral bargaining protocol. One defines the states and state transitions of the buyer (Figure 5-1) and the other one the supplier (Figure 5-2).

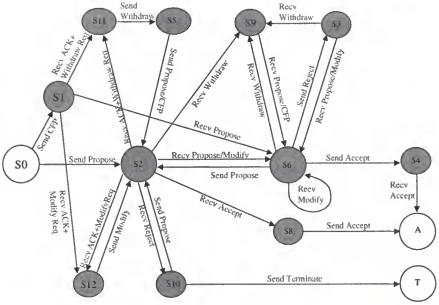

Figure 5-1: Buyer's Negotiation Protocol State Transition Diagram

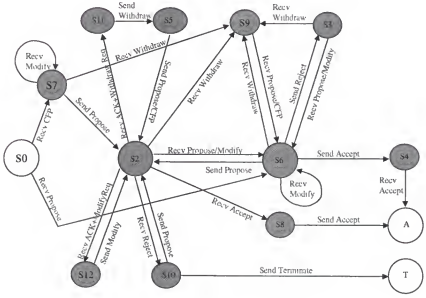

Figure 5-2: Supplier's Negotiation Protocol State Transition Diagram

In the above two diagrams, there are a total of 14 different negotiation states that a negotiation server can be in during a bilateral bargaining process. Since a negotiation server can be the initiator of a negotiation transaction and, at the same time, the counterpart of a transaction initiated by another server, the transition diagrams define the various states that a server can be in when playing both roles. We shall use "buyer" and "supplier" to represent these two roles. The meanings of these states are shown in Table 5-2.

Table 5-2 Negotiation State Semantics

S0	Initial State
S1	CFP is sent
S2	Propose is sent
S3	Reject is sent
S4	Accept is sent
S5	Withdraw is sent
S6	Propose/CFP/Modify is received
S7	CFP/Modify is received
S8	Accept is received
S9	Withdraw is received
S10	Reject is received
S11	Received the Acknowledge of the Propose/CFP that was send and the user requests the Withdraw of that Propose/CFP
S12	Received the Acknowledge of the Propose/CFP that was send

	and the user requests the Modify of that Propose/CFP
A	Agreement is reached
T	Negotiation is unilaterally terminated

S0 is the initial state. S1 is entered after the buyer sends out a CFP. S7 is entered when the supplier receives a CFP. The remaining states are shared by the buyer and the supplier. In general, a negotiation transaction is initiated by the buyer after sending out a CFP (Call-for-Proposal). In the CFP, the buyer indicates its interest to start a negotiation on some negotiation item(s) and provide some negotiation conditions and constraints for the negotiation item(s). To respond to a CFP, the supplier would send a Propose message (including the proposal) and provide the negotiation conditions and constraints from its own perspective. Then the buyer can decide if the conditions and constraints in the proposal are acceptable or not. If they are not acceptable, further negotiation needs to be conducted to resolve the differences by sending a Propose message (i.e., counterproposal) to the supplier. In another case, if the buyer knows precisely the conditions and constraints of a negotiation item it wants to send to a supplier, it can use Propose to start the negotiation instead of CFP. Hence, the supplier may also receive a Propose to start a negotiation. Either case (using CFP or Propose to start the negotiation) is defined by the messages and state transitions surrounding S0.

The most frequently used primitive used in a negotiation is Propose, which transmits a proposal or counterproposal between two negotiation servers. S2 and S6 are the two states of a negotiation server when it sends and receives Propose, respectively. To respond to a Propose, several messages are allowed: another Propose to issue a

counterproposal, a Reject to indicate the rejection of some conditions specified in the original proposal, and an Accept to indicate the acceptance of the terms and conditions specified in the previous proposal without further modifications. Note that in our negotiation protocol, a Reject message is not for specifying the termination of the negotiation. Rather, it is used to convey the dissatisfaction of some conditions to the counterpart and expect it to modify and re-send a proposal back. Thus, the Reject message contains the rejected conditions. For a negotiation server to unilaterally terminate a negotiation process for any reason, the Terminate message can be used. The state transitions and messages between state S2, S10, and T describe the cases of rejection and termination. In fact, at any state except the initial state S0, if a Terminate message is received or sent, the negotiation state would transit to state T (termination) and the negotiation process terminates. However, in order not to clutter the diagram, the state transitions to T are not shown.

To reach an agreement in a bilateral negotiation, both sides need to explicitly send Accept, the same as putting both signatures on an agreed contract. The messages from S2 to S8, and from S8 to A illustrate this case. There is another reason for using two Accept messages to finalize the agreement. Since multiple alternatives expressed by ranges and enumerations can be specified in a proposal, the first Accept may indicate several acceptable alternatives and the second Accept confirms a final accepted alternative. Henceforth, two sides can reach a definite agreement.

5.1.3 Negotiation Scenarios

To further clarify the semantics and usage of the negotiation primitives and the negotiation protocol, we shall use several example negotiation scenarios to show the meaningful state transition and message exchanges between two negotiation servers below:

(1) An agreement scenario (Figure 5-3).

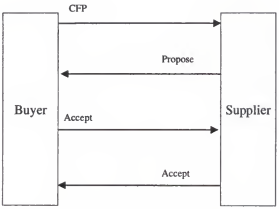

Figure 5-3: Simple Agreement Scenario

In this scenario, the buyer starts a Call-for-Proposal (CFP) (Buyer State: S0->S1) and the supplier returns a proposal (Propose) (Supplier State: S0->S7->S2) to the buyer (Buyer State: S1->S6). After evaluating the proposal, the buyer decides to Accept (Buyer State: S6->S4) the proposal and the supplier returns Accept (Supplier State: S2->S8-A) to confirm the agreement (Buyer State: S4->A).

(2) A termination scenario caused by a rejection (Figure 5-4).

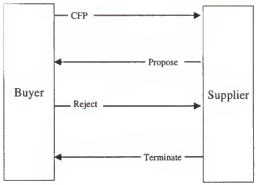

Figure 5-4: Simple Termination Scenario

Similar to scenario (1), the buyer sends CFP and the supplier returns Propose (Buyer State: S0->S1->S6, Supplier State: S0->S7->S2). After the buyer receives the proposal from the supplier, the buyer decides to Reject the proposal (Buyer State: S6->S3). After receiving the buyer's Reject (Supplier State: S2->S10) the supplier decides not to continue the negotiation and transmits a Terminate (Supplier State: S10 -> T) back to the buyer to conclude the unsuccessful negotiation (Buyer State: S3-> T).

(3) A counterproposal scenario (Figure 5-5).

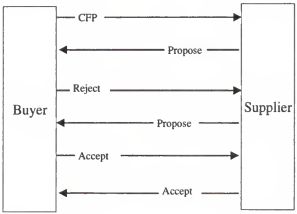

Figure 5-5: Counterproposal Scenario

Unlike scenario (2), after the supplier receives a Reject message, the supplier returns a modified proposal (Propose) back to the buyer (Buyer State: S0->S1->S6->S3->S6, Supplier State: S0->S7->S2->S10->S2). This time, the buyer is willing to accept the modified proposal (Buyer State: S6->S4->A, Supplier Sate: S2->8->A).

In a negotiation, sometimes the negotiator needs to change some conditions in the previous proposal or cancel the previous proposal that has been sent before the counterpart responds. Two negotiation primitives are provided: Modify and Withdraw. When the counterpart negotiation server receives a Modify or Withdraw message, it should stop the processing of the previous Propose message. In the case of Modify, the negotiation server needs to combine the contents of the previous proposal and the modifications specified in the received Modify message to produce a new proposal. In the case of Withdraw, the negotiation server will receive a new proposal from the sender of the Withdraw message. Both cases require the negotiation server to respond based on the new proposal. As described in the state transition diagram, Modify and Withdraw for the CFP message is similar to the case of Propose. Moreover, since the modification and withdraw of a proposal or a call-for-proposal generally requires human intervention, two human input negotiation actions, ModifyReq (for modification request) and WithdrawReq (for withdraw request), are also introduced in the protocol state diagrams. A scenario that involves Modify/Withdraw is given below:

(4) A scenario with Modify/Withdraw message exchange (Figure 5-5).

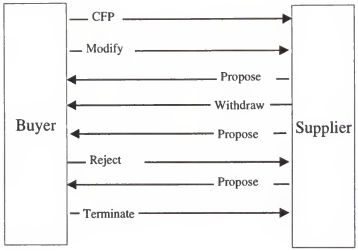

Figure 5-6: Withdraw/Modify Scenario

In this scenario, the buyer sends out a call-for-proposal (CFP) message to the supplier (Buyer State: S0->S1, Supplier State: S0->S7). Before the supplier returns any message, the buyer changes his mind and uses the Modify primitive to notify the supplier of several changes (Buyer State: S1->S12->S2, Supplier State: S7->S7). The supplier then uses the new information to generate a proposal and sends it back to the buyer (Supplier State: S7->S2, Buyer State: S2->S6). Before the buyer responds, the supplier also changes his mind. The supplier then uses the Withdraw message to first notify the buyer of his intention and then sends a new proposal (Propose) to the buyer (Supplier State: S2->S11->S5->S2, Buyer State: S6->S9->S6). The buyer rejects the new proposal (Reject) and the supplier sends another counterproposal (Propose) (Buyer State: S6->S3->S6, Supplier State: S2->S10->S2). The buyer terminates the negotiation without reaching an agreement. (Buyer State: S6->T, Supplier State: S2->T).

5.1.4 Synchronization Issue in Using Modify and Withdraw Primitives

The above scenario is an uncomplicated case in using Modify/Withdraw in a negotiation. Unlike other negotiation primitives designed to respond to messages received from the counterpart, Modify and Withdraw primitives are designed to change the proposal that was sent out by a negotiation server. Sometimes, this may cause synchronization problems between the two negotiation servers. If we call the negotiation server that sends a Modify/Withdraw as S and the receiver as R, it is possible that R may have responded to the previous proposal before it receives the Modify/Withdraw message (Synchronization Problem 1). To solve this problem, as shown in Figure 5-1 and Figure 5-2, several state transitions have been added to the state transition diagrams so that the receiver (R) of Modify/Withdraw can reverse the state to process Modify/Withdraw, even if it has responded to the old proposal. Correspondingly, S needs to discard the obsolete message from R and wait for its new message. This can be achieved by letting the negotiation server transit to some states (e.g., S5, S11, S12) in which all the incoming messages will be ignored, including the obsolete response message for the old proposal after it receives the Modify/Withdraw request from a human.

In another case, the negotiation server S and R may send Modify or Withdraw messages at the same time after S sends a proposal and R returns a counterproposal. However, it is obvious that only S's Modify or Withdraw can take effect because R's counterproposal has been invalidated by S's Modify or Withdraw of its proposal. Therefore, the Modify or Withdraw of R's counterproposal should be invalid also (Synchronization Problem 2). To solve this problem, we design a mechanism to prevent both sides from sending Modify/Withdraw at the same time. In other words, Modify/Withdraw is similar to an exclusive token that only one negotiation server can use

at any time. The mechanism works as follows. First, after a negotiation server receives a Propose, an Acknowledge message will be sent if the Propose message will be processed instead of being ignored (To simplify the diagrams, we do not show the sending and receiving of Acknowledge message). Second, a negotiation server can send a Modify or Withdraw message only after its proposal has been acknowledged. In this case, after the user has begun the Modify/Withdraw process by sending a ModifyReq/WithdrawReq request to the negotiation server, all the incoming obsolete messages will not be acknowledged. Therefore, the counterpart can not send its Modify/Withdraw message before it processes the Modify/Withdraw message. We shall give more details to explain our solution using the following example.

(5) A more complex scenario using Modify/Withdraw primitive (Figure 5-7).

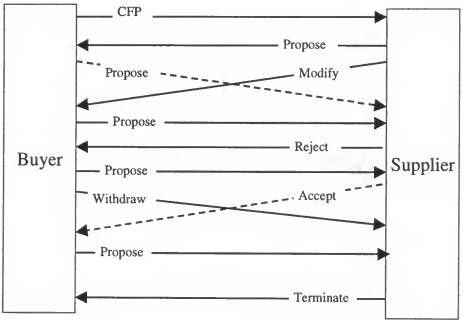

Figure 5-7: Complex Withdraw/Modify Scenario

In this scenario, the buyer sends out a call-for-proposal (CFP) message to the supplier and the supplier replies by a Propose (Buyer State: S0->S1-S6, Supplier State: S0->S7->S2). After the supplier sends out the Propose, the supplier changes his mind and initiates a modification request (ModifyReq) (Supplier State: S2->S12). If at this moment the buyer has replied to the original proposal by sending out a Propose (Buyer State: S6->S2), this message will not be processed by the supplier since the supplier has transited to state S12 after starting the modification action and will thus ignore all the messages received. This result is correct because the Propose message from the buyer has become obsolete due to the supplier's modification of its proposal (solved the Synchronization Problem 1). Moreover, at this moment, if the buyer wants to do a modification on or withdraw of the obsolete proposal, the permission will not be granted because the Propose has been ignored by the supplier and the buyer will not receive the Acknowledge from the supplier (solved the Synchronization Problem 2).

We can achieve the synchronization of the Withdraw primitive using the same mechanism. For example, in the subsequent negotiation steps, the supplier sends out the Modify (Supplier State: S12->S2). The buyer receives it, processes it, and sends out a Propose (Buyer State: S3->S6->S2). Next, the supplier rejects the proposal and the buyer sends another proposal (Supplier State: S2->S6->S3->S6, Buyer State: S2->S10->S2). After the buyer sends the proposal, if the buyer changes his mind and decides to compose a new proposal to replace the old one, the buyer will first send out a Withdraw (Buyer State: S2->S11->S5). However, before the Withdraw arrives on the supplier side, the supplier may have accepted the previous proposal (Supplier State: S6->S4). In this case, after receiving the Withdraw from the buyer, the supplier knows that the previous Accept

is an obsolete response message. The supplier then goes back to the state S9 and wait for a new proposal from the buyer (Supplier State: S4-> S9). At the buyer side, the buyer ignores the obsolete Accept message and sends out a new proposal (Buyer State: S5->S2). After the supplier receives the new proposal (Supplier State: S9->S6), the supplier decides to terminate the negotiation (Supplier State: S6->T, Buyer State: S2->T) and the negotiation process finishes without any agreement.

In summary, the negotiation primitives and the negotiation state transition diagrams formally define the message types and the rules for conducting the message exchanges between negotiation servers. They form the basis for the negotiation servers to carry out the negotiation process automatically.

5.2 Negotiation Message and Negotiation BOD

During negotiation, the messages exchanged between negotiators contain not only the primitives but also the data and constraint specifications relevant to these primitives. For example, CFP, Propose, and Accept messages would each contain a proposal that specifies the data conditions and constraints of a negotiation item(s). In some existing negotiation systems, constant values can only be given to data attributes and no constraint can be specified. This does not give a sufficient expressive power for proposal specifications. In our work, we use the same content specification language (OOCSL) that is used for registering users' requirements and constraints to specify proposals and counterproposals. The following example shows a sample proposal issued by the buyer's negotiation server during the negotiation of a computer purchase.

```
Entity Computer_System{

ATTRIBUTE_CONSTRAINT:
      model          String        EQUAL "PentiumII 350"
      monitor        Integer       ENUMERATION[15, 17, 19]
      memory         Integer       RANGE[16m..64m]
      hard_drive     Integer       RANGE [1g.. 12g]
      unit_price     Float         EQUAL 1700.00}
      deliver_day    Integer       RANGE[3..12]
      quantity       Integer       RANGE[300..400]

INTER-ATTRIBUTE-CONSTRAINT:
 Constraint1     monitor = 15 implies unit_price < 1500
}
```

Abstractly speaking, a proposal can be viewed as a hierarchical structure consisting of attributes and constraints. A composite attribute is represented by a substructure of attributes and their constraints. To automatically generate a new proposal, the registered constraints can be used, however, without the processing information, such as the priority of constraint evaluation. In another case, the outgoing proposal can be different from the original registered requirements. For instance, a negotiation server may want to specify stronger conditions in the proposal than what are specified in the original registration information, if doing so is more advantage to the users it represents.

To ensure the interoperability between negotiation systems, a set of XML-based Business Object Documents (BODs) have been designed and implemented to wrap the data contents and constraints associated with various negotiation primitives. The standardized semi-structure format makes the transmission and processing of negotiation messages over the Internet easier. Corresponding to the negotiation primitives given in Section 5.1, the following eight Negotiation BODs are defined. The name of each BOD consists of a verb which describes an action, followed by a noun which describes the

object on which the verb acts. The verbs of our BODs are: CALLFOR, PROPOSE, ACCEPT, MODIFY, TERMINATE, WITHDRAW, REJECT, ACKNOWLEDGE. The noun of the BODs are PROPOSAL except TERMINATE, which uses NEGOTIATION as the noun. The sturcture of the Propose Proposal BOD is shown below:

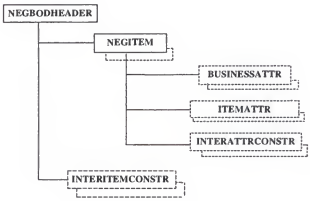

Figure 5-8: Structure of Proposal Proposal BOD

For each negotiation BOD, we also define a corresponding XML DTD (Data Type Definition) file to constrain the physical transmitted data format. Appendix B shows the DTD for the Propose Proposal BOD and Appendix C shows the XML BOD message for the sample computer purchase proposal given before.

5.3 Negotiation Decision Process

The above sections described the negotiation primitives, protocol and message structures needed for negotiation servers to communicate with each other. We now turn our attention to the decision process each server takes to carry out automated negotiations. We shall show how a negotiation server utilizes the registered negotiation

knowledge and information to arrive at the decision of acceptance or rejection, evaluate different alternatives, and generate counterproposals automatically. In this dissertation, we focus on three main activities of the negotiation decision-making process: namely proposal evaluation, conflict resolution, and selection of alternative negotiation conditions.

In general, the negotiation process consists of two phases. First, finding the intersection area between requirements and constraints specification of two negotiators through the process of conflict resolution and constraint relaxation. Second, finalizing a final mutually acceptable agreement among the alternatives found in the intersection area.

In the intersection search stage, the negotiation system checks to see if the received proposal can satisfy its own requirements (specified in the registration information), and then decides to either accept or reject the proposal. The evaluation process can be treated as a Constraint Satisfaction Problem (CSP) [TSA93] as follows. The variables are the negotiation attributes. The constraints are the combination of the constraints defined in an incoming proposal and the registered negotiation requirements of the server that evaluates the proposal. The task of the CSP engine is to find alternative conditions and constraints that satisfy both sides (bilateral bargaining).

If we treat each negotiation attribute as a dimension in a multi-dimensional negotiation space, the constraint-based negotiation requirements would define some areas or points in the negotiation space. For instance, in a bilateral bargaining situation, if the negotiation requirements and constraints of two negotiators are defined over two negotiation attributes: price and delivery_day. They can be represented by the two areas shown in the Figure 5-9.

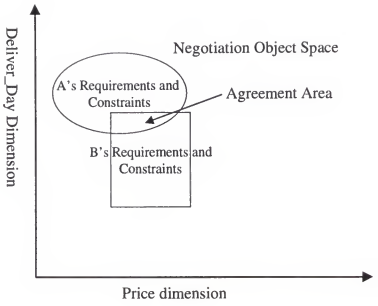

Figure 5-9: Negotiation Requirement and Negotiation Space

The negotiation process can then be defined as "negotiators jointly searching a multi-dimensional space and then agreeing to a single point in the space" [OLI96]. As we can see in the diagram, any point in the intersection area can be a possible agreement because it is a necessary condition that the agreement must satisfy the requirement constraints of both negotiation parties. However, such an intersection area may not exist at the beginning of a negotiation process. It is through a process of identifying conflicts of constraints and the resolution of conflicts (e.g., by concessions) that a final agreement can be established.

According to Lewicki [LEW97], the negotiation process is "the decision-making process of resolving a conflict involving two or more parties over multiple interdependent, but non-mutually exclusive goals." As we can see in the above section, conflicts may be identified when a negotiation server evaluates the proposal against some

registered requirements and constraints of a user or business registration. In order to continue the negotiation process, when conflicts have been identified, the negotiation server needs to take actions to resolve conflicts, e.g., by modifying data conditions and/or constraints to suit the other side (i.e., concession), asking for human intervention, rejecting a proposal with an explanation of reasons for rejection, etc. The result of these actions will become the basis for generating a response, e.g., a counterproposal message, a rejection message, etc. The rules used for conflict resolution action are called negotiation strategies. Unlike the requirements and constraints that are revealed to the counterpart through proposal exchanges, we believe that negotiation strategies should be kept private since they are vital to protect the interest of a negotiator and to control the direction of the negotiation process.

Concession rules can be applied to modify the requirements and/or constraints of a negotiation item in order to resolve some conflicts. For example, by reducing the price and/or shorten the delivery days, the price and delivery date requirements of the counterpart can be satisfied. Basically, modifying the constraints means that negotiators are searching new alternatives in the negotiation space because, according to Figure 5-7, changing the constraints means that a negotiator is changeing his requirement area in the negotiation space. When multiple conflicting constraints exist, it is not necessary to resolve all of them in one round of message exchange. Instead, the conflicting constraints may be resolved one or a few at a time, depending on the negotiation strategy used and the nature of the conflicts. Therefore, the resolution process may extend into several rounds of message exchanges between negotiation servers.

After negotiation constraints have been established to be consistent, the negotiation servers need to decide on the trade-off between different acceptable alternatives. This is because that the intersection area may have multiple agreement points. The servers need to make a selection among the alternatives as their final agreement. Normally, a negotiation server should pick an alternative that is most beneficial to the client it represents. However, an alternative most beneficial to one side may not be beneficial enough to the other side (e.g., above a certain acceptance level). How and when to search for another alternative and how and when to compromise by reducing the acceptance threshold are the main concerns in this phase of the negotiation. Figure 5-10 shows the flow of the overall negotiation decision process.

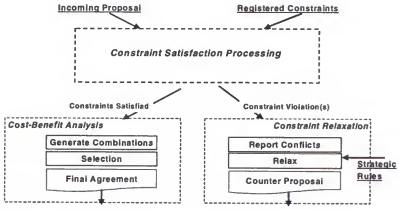

Figure 5-10: Negotiation Decision Process

The negotiation server first evaluates the proposal against the registered requirements and constraints. If the constraints do not have any conflict, the negotiation server would conduct the selection process to decide the final agreement. Otherwise, it

will activate strategic rules to resolve the conflicts. The result from either branch forms the basis of the response message.

When the response message arrives at the counterpart's negotiation server, a similar process would take place to generate its response message. The process would continue automatically until an agreement is found or either one of the negotiation server unilaterally terminates the process. The following two sections will describe two key techniques used by the negotiation server to support the automatic negotiation decision process: Constraint Satisfaction Processing and Event-Trigger-Rule Processing.

5.4 Negotiation Constraint Satisfaction Processing

As stated previously, negotiation proposal evaluation can be treated as a constraint satisfaction problem. Therefore, in a negotiation server, a constraint satisfaction processing (CSP) engine needs to be implemented to perform the proposal evaluation when the negotiation server receives a proposal. In the evaluation process, the data, the attribute constraints, and the inter-attribute constraints specified in the proposal are evaluated against the registered data and constraints of the recipient.

In the evaluation process, attribute constraints are evaluated before inter-attribute constraints. The evaluation of attribute constraints follows the priority order of the registered information as explained in Section 4.1. Comparison operators for Range and Enumeration are introduced. If a conflict is found in an attribute constraint during this process, the corresponding event is posted to trigger the evaluation of an event history and the associated negotiation strategies. The negotiation strategies may automatically relax the violated constraint (e.g., to reduce the delivery date by 2 days to meet or to come

closer to the buyer's desired delivery date) or call for human intervention to relax the constraint. The relaxed value can be used directly in the counterproposal to be sent to the negotiation server of the other party. In this case, the client's strategy is to "give in a little bit (i.e., one violation) at a time" in an attempt to meet the buyer somewhere in the middle. Alternatively, the evaluation of attributes and attribute constraints can continue until a pre-specified number of conflicts are found. The latter approach may speed up the negotiation process by reducing the number of message exchanges but may not necessarily result in an optimal result for the client because some constraints may be relaxed unnecessarily.

If no conflict is found during the attribute constraint evaluation or after the relaxation (meaning, for each attribute, the range or enumeration of values of the received proposal overlaps with those of the registered data), a set of alternative acceptable data conditions may exist. Each of these alternatives represents a negotiation item description that satisfies the attribute constraints of both negotiation parties. Next, the alternatives are evaluated against the inter-attribute constraints that come with the proposal to filter out those alternatives that do not satisfy the inter-attribute constraints. The remaining ones are then evaluated against the registered inter-attribute constraints of the client by following its own priority order.

In the traditional CSP algorithm, when the inter-attribute constraints are evaluated, all the constant-value-based combinations that can satisfy the attribute constraints need to be generated and tested against the inter-attribute constraints. This approach may generate a large number of instances. For example, if there are four Integer type negotiation attributes and each attribute constraint defines a range having 1000 acceptable

values, e.g., [1000 .. 2000], [5000 .. 6000]. The total number of instances generated will be 1000*1000*1000*1000=1 trillion. Obviously, testing these instances against the inter-attribute constraints will consume a huge amount of computing resource.

To improve the CSP engine performance, we design an algorithm that can generate instances based on interval value instead of constant value for each attribute. The interval defines the upper and lower bound values of the attribute. It uses the symbols "(" and ")" to indicate that the boundary values should be exclusive and the symbols "[" and "]" to indicate that the boundary values should be inclusive. All the attribute constraints can be represented by one or more such intervals. For instance, the constraint "EQUAL a" can be defined as "[a,a]", "ENUMERATION[a, b, c]" can be defined as "[a,a], [b,b], [c,c]", and "RANGE[a..b]" can be mapped to interval "[a, b]". In this case, after the attribute constraint checking, the result will be a set of intervals for each attribute. We can then generate the instances based on the attribute intervals. Since each interval may include multiple constant values, the number of instances generated by this approach would be much less than the one generated by the constant-value-based approach. To distinguish the instances of the constant-value-based approach from those of instance of the interval-based approach, we call the latter interval records.

However, if we generate records right after the attribute constraint checking and treat each interval record as a collection of multiple constant-value-based instances, it is possible that some instances of the record may produce a TRUE value with respect to an inter-attribute constraint, the rest of instances of the record may produce a FALSE value. In this case, we can not assign a TRUE or FALSE value to the interval record with respect to the inter-attribute constraint. Therefore, we have to split the interval record

into two or more records so that each new record can have its own evaluation result. The interval splitting algorithm is described below.

In an inter-attribute constraint, each atomic predicate, which contains a relational operators (=, !=, >, <, >= or <=) can be represented by one or more attribute intervals. For example, x >=100 can be represented by the interval [100, POSITIVE_INFINITE). To distinguish the attribute intervals derived from an inter-attribute constraint from intervals derived from the attribute constraint checking process, we call the latter an attribute constraint interval. In the evaluation, if an attribute constraint interval overlaps with the attribute interval derived from an atomic predicate of an inter-attribute constraint, we split the attribute constraint interval into two or more intervals. For example, an attribute constraint interval for attribute x is Ax. An attribute interval derived from a predicate of x in the inter-attribute constraint is Bx. When checking the inter-attribute constraint, we create two new intervals for the attribute x. The first interval is [Ax $\cap$ Bx], and the other interval is [Ax - (Ax $\cap$ Bx)]. In fact, the latter formula may produce two intervals. The same task of interval splitting needs to be performed for all other attributes that are used in the predicates of the inter-attribute constraint, e.g., attributes y and z. We repeat the above procedure to check the attribute constraint intervals against other inter-attribute constraints. As a result, more new intervals may be produced for each attribute. Finally, we can generate interval records based on all the combinations of new intervals for all the attributes (e.g., x, y, and z). We shall illustrate the process using the following example. Assume that the attribute-constraints and inter-attribute constraints after the attribute constraint checking process are:

```
X: RANGE [10 .. 110]
Y: RANGE [300 .. 600]
```

```
IAC1:  X > 50 --> Y > 400
```

IAC2: X < 70 --> Y < 50

In the traditional algorithm, a total of 100*300 = 30,000 instances will be generated and tested against the two inter-attribute constraints. In our algorithm, the initial attribute constraint intervals are:

```
{
 X: [10, 100]
 Y: [300, 600]
}
```

First, we evaluate it against the inter-attribute constraint IAC1, the interval of attribute X is split into two:

```
{
 X: [10, 50], (50, 110]
 Y: [300, 600]
      }
```

The interval of Y needs to be split also:

```
{
 X: [10, 50], (50, 110]
 Y: [300, 400], (400, 600]
}
```

Then we evaluate the result against IAC2, the intervals are split further as shown below:

```
{
 X: [10, 50], (50, 70), [70, 110]
 Y: [300, 400], (400, 500), [500, 600]
      }
```

After we finish the interval splitting process, we can generate interval records based on these new intervals as shown in Table 5-3. After we generate the records, we can derive the evaluation result for each record against the inter-attribute constraints. If the evaluation result for any inter-attribute constraint is FALSE, the record should be removed. For example, as shown in the above table, we can remove records 3, 4, 6, and

7. The remaining records are the records accepted by the CSP engine based on the input attribute and inter-attribute constraints.

Table 5-3 Interval based Records

Record Number	X	Y	IAC1 Result	IAC2 Result	Satisfied
1	[10..50]	[300..400]	T	T	Yes
2	[10 .. 50]	(400..500)	T	T	Yes
3	[10 ..50]	[500..600]	T	F	No
4	(50..70)	[300..400]	F	T	No
5	(50..70)	(400..500)	T	T	Yes
6	(50..70)	[500..600]	T	F	No
7	[70..110]	[300..400]	F	T	No
8	[70..110]	(400..500)	T	T	Yes
9	[70..110]	[500..600]	T	T	Yes

As we can see, because each interval covers multiple attribute values, the number of generated records is much smaller than the number of constant value instances. In the above example, only 9 records are generated as compared to the 30,000 constant-value-based instances generated by a traditional CSP engine. Therefore, the CSP engine's performance is significantly improved.

After the inter-attribute constraint evaluation is completed, if no violation is found (meaning, there exists at least one negotiation item instance that satisfies all the inter-attribute constraints), the proposal is acceptable to the client. In that case, an Accept

message is sent to the other party. In the event that multiple negotiation item instances satisfy all the constraints of the client, the user-configured CBADM is invoked to quantitatively evaluate the alternatives. The negotiation item instance that is most beneficial to the client is selected for acceptance. If a violation of any inter-attribute constraint is found (i.e., no instance satisfies the constraint), an event is posted to activate some strategic rules to either automatically relax the constraint or call for human intervention.

5.5 Event-Trigger Rule Server

In the automated negotiation process, the Event-Trigger-Rule (ETR) server is the key component for managing and automatically triggering the negotiation rules to resolve negotiation conflicts when conflicts are detected. Specifically, the ETR server is responsible for receiving events from the negotiation CSP engine. Each event may trigger multiple strategic/tactic rules to relax the violated constraints or to call for human intervention.

The execution of a negotiation strategic rule is based on a dynamic compilation approach which is faster and more efficient than an interpretation approach used by most rule systems. For each rule defined at the build-time, the corresponding Java rule code is generated, complied and stored in the ETR server rule base based on the rule specification. At runtime, the ETR server provides the following services to enable the automatic execution of negotiation strategic rules:

(1) Receive and analyze events

(2) Interpret the trigger specification

(3) Load and execute the corresponding rule code

To allow new rules to be added and/or be modified during the negotiation process, the ETR server uses the technique of "dynamic loading" by translating the new and/or modified rules into Java classes at run-time and using a class loading and/or reloading facility to load the new rules and/or to replace the old rule classes by the new ones at runtime. The dynamic rule processing capability provided by the ETR server is particularly useful since business and negotiation constraints and policies may change during a negotiation process. It allows a negotiation client to have more flexibility in controlling the negotiation process.

The executable code for the strategic rule SR1 is shown in Appendix D. In the sample code, each rule corresponds to a class. Different parts of the rule specification are translated into different methods of the class. The method of notify_Event() is the triggering interface and expresses the overall logic of the rule specification. The rule is activated when its event is posted. The relationship between the triggering and the rule is specified in a trigger specification.

The runtime ETR server architecture is given in Figure 5-11.

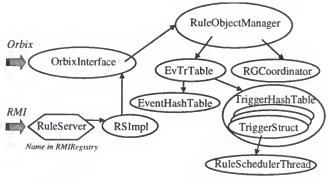

Figure 5-11: ETR Server Architecture

As we can see in the diagram, the ETR server is implemented on top of the Internet infrastructure in which the ETR server can handle the events posted either by Orbix, an standard Object Request Brokerage (ORB) product of the Iona Enterprises, or by Remote Method Invocation (RMI), the Java Remote Object Invocation Standard. During a negotiation process, the RMI interface of the ETR server is used to interact with other components (e.g. CSP engine) of the negotiation server.

5.6 Example Scenarios

We shall further explain the automated negotiation execution process by using our previous computer purchase example. The buyer's proposal given in Section 4.4 is processed by the negotiation server NSB against the supplier's registration information given in Section 4.1. First, attribute-constraints are checked. According to the priority values, the **NONNEGOTIABLE** constraint for quantity is checked first. NSB finds that the user's quantity constraint is satisfied. Then, the delivery_day constraint is checked and NSB finds a violation since the supplier's constraint on delivery_day is in the range of [14..21], whereas the buyer requires a delivery day in the range of [3..10]. As a result, NSB posts a violation event " SupplierComputer_Systemdelivery_day" and triggers rule SR1 to relax the Supplier's delivery_day constraint. According to the action part of the rule, NSB changes the supplier's deliver_day constraint from RANGE [14..21] to RANGE [12..21].

No other constraints are violated in the example and the outgoing counterproposal to the buyer is as follows. Note: the outgoing counterproposal does not contain priorities since the buyer most likely has his own constraint evaluation order.

```
ENTITY Computer_System {
```

```
ATTRIBUTE-CONSTRAINT:
      model          String        EQUAL "PentiumII350"
      memory         Integer       ENUMERATION[32m,64m, 96m]
      monitor        Integer       EQUAL 17
      hard_drive     Integer       ENUMERATION [4g, 6g, 8g]
      unit_price     Float         EQUAL 1850.00
      deliver_day    Integer       RANGE[12..21]
      quantity       Integer       RANGE [300..400]

INTER_ATTRIBUTE_CONSTRAINT:
      iac1   quantity >= 400 implies deliver_day >= 16
      iac2    memory < 64m implies hard_dirve < 4g
      }
```

When the counterproposal arrives at the buyer's NSA, it will go through the same processing procedure except that the registered constraints, strategic rules, and the cost-benefit evaluation model of the buyer are used. According to the buyer's requirements and constraints defined in Section 4.1, the delivery_day constraint is checked first and a violation is found because the buyer requires the delivery_day in the range of [3..10] but the supplier can only promise a delivery_day in the range of [12..21]. As a result, NSA posts a violation event " BuyerComputer_Systemdelivery_day" and triggers rule BR2 to relax the Buyer's `delivery_day` constraint. According to the action part of the rule, NSA changes the buyer's `deliver_day` constraint from RANGE [3..10] to RANGE [3..12]. After the relaxation, the delivery_day constraints of the supplier and the buyer are consistent. NSA would continue to check other constraints, including inter-attribute constraints. After the constraint evaluation, the NSA may decide to reject or accept the counterproposal, or generate another counterproposal to be sent back to NSB. This back-and-forth process continues until a mutual agreement is reached or either side terminates the negotiation process.

CHAPTER 6
SYSTEM DESIGN AND IMPLEMENTATION

This chapter presents the design and implementation of the prototyped Web-based Automated Negotiation Server (WANS) using the methodologies and techniques presented in the previous chapters. Section 6.1 gives the overall architecture of WANS and Section 6.2 presents our efforts on the user interface to enhance the usability of the system. Section 6.3 describes an integrated demonstration scenario that uses the negotiation server in the context of a supply chain management system.

6.1 System Overview

The Web-based Automated Negotiation Server (WANS) consists of the components shown in Figure 6-1. It is designed to support bilateral, multi-issue, bargaining-type negotiations over the Internet. The prototype system is designed and implemented based on the object-oriented techniques. All the components are implemented as objects. To increase the system portability, Java is used as the only programming language. In addition, to make all the services of WANS be accessed through the Internet without client software installation and configuration, Web programming techniques, e.g., dynamic html, Servlet, and Applet, are adopted.

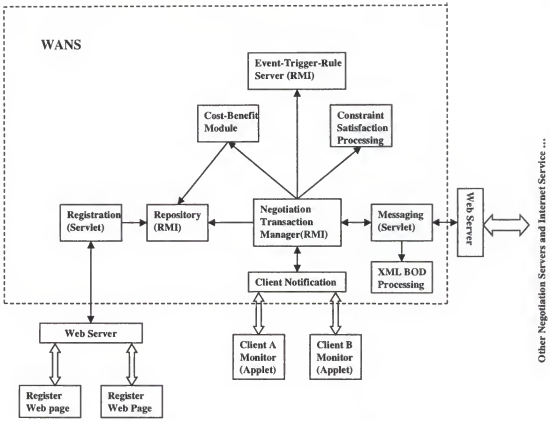

Figure 6-1: WANS System Components

We shall describe the functionality and implementation technique for each component.

(1) RegisterWebPage is a dynamic HTML-based GUI for users to register negotiation knowledge, including requirements, negotiation strategies, and the data to be used in the cost-benefit evaluation model.

(2) Registration is a Java servlet program that can dynamically generate the Web pages that make up the registration process based on the definition of negotiation items

(i.e., the ontology). The servlet also stores the user registration information and negotiation knowledge in the negotiation Repository.

(3) Repository provides persistent data service for the negotiation server. The repository consists of two modules, which are not shown in Figure 6-1. First, the MetaDataManager stores and manages the ontology information for each negotiation item (e.g., a product or a service). The MetaDataManager is mainly used during build time for providing ontological supports to Registration and ETRServer. Second, NegotiationInformationBase stores and manages the negotiation knowledge, state, and transaction information during build-time and run-time negotiation processes. The Repository service is implemented based on Java Remote Method Invocation (RMI) and Serialization techniques.

(4) ClientConsole is an applet-based GUI used to let users monitor the ongoing negotiation activities and to intervene in the negotiation process when necessary. ClientConsole simplifies the use of WANS significantly. We shall further elaborate on our efforts in building GUIs for human-computer interactions in Section 6.2.

(5) ClientNotification is a socket-based communication channel between the NegotiationTransactionManager and the ClientConsole.

(6) CostBenefitModule provides cost-benefit analysis and evaluation capabilities for the negotiation system to make a selection from multiple negotiation alternatives. Based on the evaluation scores, two functions are provided: (1) Return the best alternative, and (2) return all the alternatives sorted by preference scores.

(7) ConstraintSatisfactionProcessing is responsible for evaluating the constraints specified in the incoming negotiation proposal or counterproposal and the pre-registered

constraints and requirements. As introduced in Section 5.4, this module implements an efficient constraint satisfaction processing (CSP) algorithm so that the constraints can be evaluated in a more efficient and effective way than the traditional CSP algorithm. The result of the CSP evaluation servers as the basis for the negotiation system to trigger the corresponding negotiation strategic rules to relax constraints (in the case of violations) or to make a selection from multiple alternatives that satisfy the constraint evaluation.

(8) ETRServer supports the dynamic triggering and execution of negotiation rules. The ETR server was developed by our colleagues in the Database Systems Research and Development Center.

(9) Messaging is the communication interface between negotiation servers on the Internet. Currently, it is implemented based on the "subscribe/publish" Internet messaging.

(10) XMLBODProcessing wraps and unwraps messages packaged as XML BOD to ensure the interoperability between different negotiation systems.

(11) NegotiationTransactionManager is the coordinator of the negotiation server to manage all the different components to automatically carry out the negotiation tasks by following the states and transitions of the negotiation protocol defined in Section 5.1.

To further elaborate on the role of each component at build-time and run-time of the negotiation process, we shall describe a complete automated negotiation process by referencing the activity of each component.

The negotiation process starts with users' or business organizations' registrations through the Web-based graphical interfaces (RegisterWebPage). They can enter requirements, constraints, cost-benefit model configurations, and strategic rules. All the

information is stored in the Repository. At run-time, a user starts a negotiation transaction through the ClientConsole. After the NegotiationTransactionManager receives this request, it will initiate a new transaction for this user. The NegotiationTransactionManager retrieves the necessary registration information from the Repository, forms a call-for-proposal (CFP), and transmits it to the designated negotiation server of the counterpart using the Messaging infrastructure. The Messaging infrastructure will call the XMLBODProcessing to pack the message in XML BOD format before sending the CFP.

On the other side, the CFP will be received by Messaging module. After the CFP is parsed by the XMLBODProcessing component, the CFP will be passed to the NegotiationTransactionManager, which will call the ConstraintSatisfactionProcessing to evaluate the CFP. If there are violations, the ETRserver will be activated to resolve the violations. This may lead to sending a counterproposal with modified constraints or to sending a rejection. If all constraints are satisfied and an intersection is found, the CostBenefitModule will be called upon to find the best alternative for the final agreement. In this case, an acceptance will be sent out. The response message will again be packaged in XML BOD format and returned through the Messaging component. Before the message is sent out, the negotiation constraints, state, together with other negotiation transaction information (transactionID, counterpart negotiation server name, etc) will be stored in the Repository. When the response to the Sent message is received, the transaction information can be retrieved from the Repository so that the negotiation server can continue the negotiation from where it previously left off.

The entire negotiation process may go through several rounds of proposal exchanges until an agreement or disagreement is reached. In either case, the NegotiationTransactionManager will remove the transaction information stored in the Repository. All the important steps can be monitored by the user through the ClientConsole.

During the design and implementation of WANS, attentions are also paid to enhance the system scalability so that the negotiation services can be efficiently shared by multiple users via the Internet. First, WANS uses multi-threaded programming to support multiple users to conduct multiple concurrent negotiation processes. Second, the negotiation server can be replicated at multiple Web sites. Finally, the main components of WANS can also be installed in a distributed environment. For example, the NegotiationTransactionManager, the ETR Server, and the Repository can be installed on different machines connected to the Internet. Therefore, the components can cooperate via the Internet to carry out the negotiation tasks. As a result of the workload distribution, system performance and scalability are improved.

6.2 Enhance the Usability of Negotiation Server

Ease of use is another important objectives when designing and implementing WANS. WANS provides a set of GUIs to assist human users in using the negotiation service both during build-time (negotiation information registration) and run-time (negotiation execution). We have described the registration interfaces for negotiation requirements, strategies, and cost-benefit models in Chapter 4. Another important

interface, the run-time negotiation console, is used for human monitoring and intervention purposes.

In a complex negotiation process, a negotiation server may not be able to carry out the negotiation process fully automatically due to the lack of pre-registered negotiation rules to handle different situations. Human monitoring and intervention is necessary to ensure that the negotiation process can come to a proper termination and the negotiation result is what a user or organization desires. For this purpose, the negotiation server provides a ClientConsole (shown in Figure 6-2), which is an applet-based graphical user interface (GUI) to allow human users to monitor the negotiation process and to intervene by overwriting the results produced by their negotiation servers or by providing the needed information to the servers when necessary.

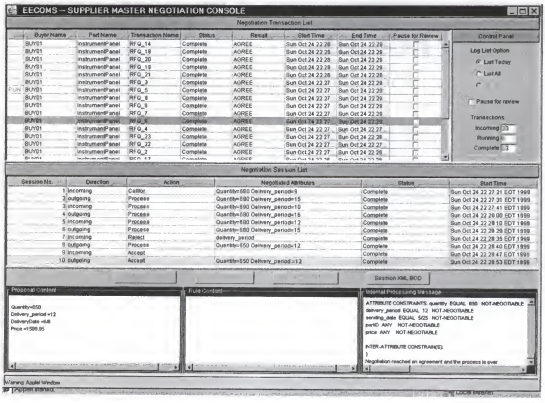

Figure 6-2: Negotiation Run-time Console

The information presented in the ClintConsole includes output messages related to the ongoing negotiation process such as constraint checking and rule triggering, and the proposal or counterproposal being processed. We shall explain the negotiation run-time console's functionality in more details below.

First, the top-most table inside the console window is the **Negotiation Transaction Table**, which displays all the completed as well as running negotiation transactions. General information for each transaction is displayed, including a status field that reflects its execution status: **"Running"** or **"Complete"**. A transaction consists of a number of sessions. Each session is a message exchange between the negotiation servers. By clicking on a transaction in the **Negotiation Transaction Table**, the details of its sessions are displayed in the **Negotiation Session Table** (which is shown directly underneath the transaction table).

Through the interface, users can select from two different execution modes: human confirmation mode or automatic mode. In the human confirmation mode, before the negotiation servers take any important decision steps, e.g., sending counterproposal, rejection, acceptance, etc., a confirmation click must be received from the user. A check box "Pause For Review" is available for each transaction to pause the transaction for review. If not checked, the transaction will proceed without pausing. Also, there is the same check box "Pause For Review" for each negotiation transaction. By combining the use of these check boxes, we provide a flexible way of controlling the pausing and execution of negotiation transactions. For example, if the check box on the Control Panel is checked (or unchecked), all new transactions' check boxes will automatically be checked (or unchecked). Their execution will then be paused (or not paused) after each

session. At any time, a user can check or uncheck the check box associated with a particular running transaction.

Note that when a message of a transaction is being transmitted and the check box "**Pause for Review**" for the transaction is selected, a blinking status message "**Waiting for User Response**" will be displayed. This is to remind the viewer that there is a new incoming message for the transaction that has been paused. This is particularly useful when multiple transactions are being processed concurrently. A blinking status message for those unselected transactions that have incoming messages can alert the user who may want to switch the current display to view the information of other transactions. In addition, there is a red-colored "**RUN**" indicator on the left-most column jumping among different transactions which indicates that there is a new session being executed by the transaction.

When a transaction is selected in the **Negotiation Transaction Table**, the sessions of that transaction are displayed in the Negotiation Session Table. General information for each session are shown, including the direction of the message (incoming or outgoing), the negotiation primitive (e.g., Call-for-Proposal, Propose Proposal, Accept, Reject, etc.), the attributes being negotiated, and the execution status. When a user selects (clicks on) a session, the detailed message contents, rules used to generate the contents, and XML BOD contents of the session will be displayed in the bottom text boxes. (An XML BOD window will be presented if the user clicks the button View XML BOD). If the selected session is in the paused state, the button "**Session Continue**" will be activated and awaits the user's click event after the session contents have been reviewed.

There are three additional control buttons on the **Control Panel** for selecting the display modes of negotiation transactions: "**List All**", "**List Today**" and "**List in days**". The default setting is "**List Today**", which will list the transactions that occurred today. The option "**List All**" will display a list of all the transactions, including those that have been completed in the past. The option "**List in days**" will display a list of all the transactions completed within the specified days. Furthermore, there are three transaction counters to count the received, running and completed transactions executed by the negotiation server, respectively.

Furthermore, if the user wants to modify or withdraw the message he sends out before, he can click the "**Modify**" or "**Withdraw**" button. Two additional GUIs will display to assist the user to complete the complete the composing of Modify or Withdraw messages, e.g., providing the original message contents, message format and syntax checking, verifying the changes, etc.

6.3 Integration Testing of the Negotiation Server in a Supply Chain Management Scenario

The research and development effort of our Web-based negotiation server is supported by a project funded by the Advanced Technology Program (ATP) of the National Institute of Standards and Technology (NIST), which focuses on supply chain management among participating enterprise systems using a network of heterogeneous components (i.e., ERP, application systems, agents, rule-based systems, etc.). Negotiation is a necessary activity among the many components of a supply chain and requires the interchange of data and knowledge among the negotiating parties. Our

implemented negotiation server and its supporting tools and interfaces are replicated and installed in separate computer and integrated with several other component systems of a supply chain to demonstrate the acquisition of EthernetCards from multiple suppliers. In addition to our negotiation servers, the supply chain consists of a requisition system, a supplier bid selection component, as well as a secure messaging infrastructure for communication among the individual components. The relationship between the negotiation servers and other components in the integrated system is depicted in Figure 6-3.

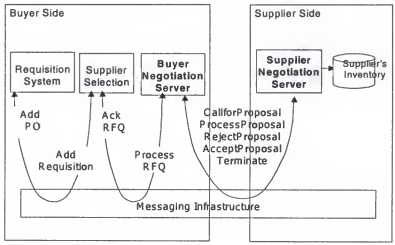

Figure 6-3: Negotiation Servers and Remaining Supply Chain Components.

A commercial requisition system on the buyer's side initiates a requisition by sending an Add-Requisition BOD to the Supplier Selection Component through the messaging infrastructure. The latter utilizes the information it maintains on regular suppliers (the suppliers that the buyer usually does business with) to derive a purchasing plan which determines how many units of the ordered item should be ordered from which

supplier and by which delivery date. A number of (supplier, quantity, delivery_date) triplets are generated to meet the demand of the buyer. In the demo scenario, the Supplier Selection Component issues a ProcessRFQ BOD (Process Request for Quote), containing one of the triplets to the negotiation server that represents the buyer. The negotiation server generates the Call-for-Proposal BOD and starts the negotiation process by sending the call-for-proposal to the negotiation server of the supplier side. Note, we are assuming here, that the negotiation server representing the buyer has identified the candidate supplier before starting the actual negotiation process.

After possibly several rounds of exchanging proposals and counterproposals, the quantity and delivery date of the original triplet may be changed due to the supplier and the buyer's constraints and rules. The price information associated with the negotiated result is retrieved from the supplier's inventory database. The negotiated result and price information is returned to the Supplier Selection Component using the AckRFQ BOD (Acknowledge Request for Quote). Three cases are possible: 1) Accept the order by providing the delivery schedule. 2) Return a counter offer with a new delivery schedule and quantities. 3) Reject the order. Thus in the case of a rejection, it is possible that the Supplier Selection Component may have to contact additional suppliers to get the number of items the buyer needs. Once the negotiations are completed, the Supplier Selection Component sends the AddPO BOD (Add Purchase Order) to the requisition system to initiate the order. In the integrated demo, there are other component systems involved. However, they are not germane to the presentation of the negotiation server and, therefore, are not included in our description.

CHAPTER 7
CONCLUSION AND FUTURE RESEARCH

7.1 Summary

In this dissertation, we have presented the design and implementation of a replicable, Web-based negotiation server for conducting bargaining-type negotiations. Specifically, we have described the underlying architecture including an object-oriented content specification language (OOCSL) for information registration, a negotiation protocol and its primitive operations, an automated negotiation process, a declarative rule-based negotiation strategy specification, and a cost–benefit decision model. The prototyped negotiation server and its supporting tools and interfaces have been tested and demonstrated in the context of a supply chain.

To enhance the replicated negotiation servers with effective and intelligent negotiation capabilities, human negotiation knowledge are captured in the following three forms: 1) requirements and constraints associated with negotiation items, 2) negotiation strategies, and 3) preference scores and aggregation functions for the cost-benefit analysis and selection of negotiation alternatives. Unlike existing approaches, in which human knowledge is hard-coded into negotiation agent programs, we use a high-level, object-oriented, declarative approach to specify human knowledge. For the negotiation requirement specification, an object-oriented content specification language (OOCSL),

which has an underlying active object model (AOM), is provided. The language and its accompanying GUI tools are used by buyers to specify the data characteristics and constraints of goods and services that they are interested in, and by sellers to publish information about the goods and services that they provide. For the specification and execution of negotiation strategies, a declarative rule language, its accompanying GUI tools, and a dynamic ETR server are provided. The constraint, event, rule and trigger specification facilities of the active object model (AOM) are extensions of the traditional object definition language and object model in which objects are defined only in terms of attributes and methods. The added semantics provided in the specifications of goods and services and negotiation proposals and counterproposals allow negotiation servers to carry out constraint verification, activation of negotiation strategic rules, and generation of counterproposals in an automated fashion. We further design an efficient constraint satisfaction processing (CSP) algorithm so that the constraints defined in the proposal and the registered requirements can be evaluated in a more efficient and effective way than the traditional CSP algorithm. The result of the CSP evaluation serves as the basis for the negotiation system to trigger the corresponding negotiation strategic rules (in the case of violations) or to make the proper selection from multiple alternatives that satisfied the constraint evaluation. For the cost-benefit analysis and selection of alternative negotiation conditions, we provide a GUI tool to allow users to tailor the cost-benefit analysis and decision model (CBADM) to meet their different needs.

Run-time consoles are also available for controlling and monitoring concurrent negotiation processes as well as for user intervention. Interoperability between negotiation servers is achieved by adopting existing standard Internet networking and

information exchange protocols, e.g., HTTP, XML, BOD and by formalizing the negotiation protocols using a set of primitives and a state transition diagram.

7.2 Future Research

Research on automated negotiation is still in its infancy. In this work, we have dealt with only a few aspects of negotiation. There are many additional research problems for further investigation. We list some of them below:

1) In this work, we have dealt with primitives and protocol for bilateral negotiations. Multi-lateral negotiations may require additional primitives and a different negotiation protocol.
2) Strategic rules implemented in this work are mainly for constraint relaxation. To support advanced negotiation scenarios, other negotiation strategies or policies may need to be incorporated. For example, a negotiation server may trade one negotiation attribute for another (e.g., give a better product attribute but increase the price), increase the amount of concession as the number of proposal-counterproposal exchanges increases, give better deal for regular customers, etc.
3) The negotiation protocol introduced in this work allows negotiation proposals and counterproposal to be exchanged between two negotiation parties an unspecified number of times until one party decides to either accept the last received proposal or unilaterally terminate the negotiation process. Obviously, it is necessary to have a time-out mechanism to ensure the termination of a negotiation process when a pre-specified time interval expires. Additionally, it is extremely useful to provide a quantitative way of measuring whether a

negotiation process is converging to an agreement or is diverging and getting into an endless exchange of proposals and counterproposals. The latter case happens when the negotiation parties continuously make conflicting requirements in the exchanges instead of making offers that can draw both parties closer to each other. Negotiation strategies and policies need to be designed and used by negotiation parties to ensure the convergence of negotiation processes. Quantitative measures on the "desirability" of negotiation proposals are important so that when one party decides to either accept or reject a proposal or to terminate a negotiation process, the decision made is quantitatively justifiable. For example, a negotiation party or the negotiation server that represents his interest can accept a proposal because its quantitative measure is above a pre-specified value, or a negotiation process is terminated because proposals received from the counterpart show a continuous decrease in desirability from a negotiator point of view.

4) Negotiation is a complex decision making problem and requires multiple types of negotiation knowledge, such as constraints, strategic rules, and cost-benefit evaluation techniques. In our current work, we assume that the negotiation expert who provides the negotiation knowledge will ensure their consistency. However, it is desired that the negotiation system can perform consistency checking automatically. This functionality is particularly important when the negotiation expert wants to update or change some negotiation knowledge, say, changing the pricing constraints from RANGE [100..200] to RANGE [150..250]. The change may cause inconsistency between the new pricing

constraint and the original strategic rules and cost-benefit model configurations related to price. In this case, the negotiation system should maintain consistency by automatically or semi-automatically modifying the corresponding negotiation strategic rules and cost-benefit evaluation model. This problem is similar to the database consistency problem. However, here we are dealing with the consistency between rules and constraints, not the database data records. Therefore, a new methodology is required.

APPENDIX A
BNF SYNTAX OF OBJECT-ORIENTED CONSTRAINT SPECIFICATION LANGUAGE (OOCSL)

```
entity ::= ENTITY IDENTIFIER BL ATTRIBUTE_CONSTRAINT COLON ac_list
INTER_ATTRIBUTE_CONSTRAINT COLON iac_list BR
                    ;

ac_list ::= ac_list ac
                    |
                    ;
ac ::= path att_type att_cons priority
                    ;
att_type ::= STRINGTYPE
                    | INTEGERTYPE
                    | FLOATTYPE
                    ;
att_cons ::= ENUMERATION LBRACKET value_list RBRACKET
                    | ANY
                    | EQUAL val
                    | DERIVED
                    | RANGE LBRACKET numeric DOT DOT numeric RBRACKET
                    ;
value_list ::= str_val_list
                    | int_val_list
                    | real_val_list
                    ;

str_val_list ::= STRING
                    |STRING COMMA str_val_list
                    ;

int_val_list ::= INTEGER
                    |INTEGER COMMA int_val_list
                    ;

real_val_list ::= FLOAT
                    |FLOAT COMMA real_val_list
                    ;
numeric ::= INTEGER
                    | FLOAT
                    ;
val ::= numeric
            | STRING
            ;

priority ::= NONNEGOTIABLE
                    | PRIORITY LBRACKET INTEGER RBRACKET
                    ;

iac_list ::= iac_list iac
                    |
```

```
                ;

iac ::= IDENTIFIER constraint IMPLIES constraint priority
                ;

constraint ::= bool_or
                |
                ;


bool_or ::= bool_or OR bool_and
                | bool_and
                ;

bool_and ::= bool_and AND bool_compare
                | bool_compare
                ;

bool_compare ::= expr EQ expr
                | expr NOTEQ expr
                | expr LESS expr
                | expr LESSEQ expr
                | expr GREATER expr
                | expr GREATEREQ expr
                | expr
                ;
expr ::= expr PLUS term
          | expr MINUS term
          | term
          ;

term ::= term TIMES factor
          | term DIVIDE factor
          | factor
          ;

factor ::= LPAREN bool_or RPAREN
          | path
          | val
          | TRUE
          | FALSE
          ;

path ::= IDENTIFIER
          | IDENTIFIER DOT path
          ;
```

APPENDIX B
PROPOSE PROPOSAL BOD XML DATA TYPE DEFINITION

```
<!ELEMENT PROPOSE_PROPOSAL_001 (CNTROLAREA, DATAAREA+)>
<!ATTLIST VERB value CDATA #FIXED "PROPOSE">
<!ATTLIST NOUN value CDATA #FIXED "PROPOSAL">
<!ATTLIST REVISION value CDATA #FIXED "001">
<!ELEMENT DATAAREA (PROPOSE_PROPOSAL)
<!ELEMENT PROPOSE_PROPOSAL (NEGBODHEADER, NEGITEM+,
INTERITEMCONSTR*)
<!ELEMENT NEGBODHEADER (NEGTRANSACTID, NEGBODID, SENDERNS,
SENDERUID, RECEIVERNS, RECEIVERUID, PROTOCOL, PROPID?,
TRANSACTDESCRIPTN? TRANSACTNAME?, PROPDESCRIPTN?,
CONSTRAINTLANGUAGE? REPLYTO?, USERAREA?)
<!ELEMENT NEGITEM(ITEMID, DESCRIPTN?, USERAREA?,
BUSINESSATTR*, ITEMATTR*, INTERATTRCONSTR*)
<!ELEMENT BUSINESSATTR(ATTRNAME, ATTRTYPE, CONSTRAINT?,
USERAREA?)>
<!ELEMENT ITEMATTR(ATTRNAME, ATTRTYPE, CONSTRAINT?, USERAREA?)>
<!ELEMENT INTERATTRCONSTR(IACNAME, CONSTRAINT?, USERAREA?)>
<!ELEMENT INTERITEMCONSTR(IICNAME, CONSTRAINT?, USERAREA?)>
```

APPENDIX C
PROPOSE PROPOSAL BOD XML MESSAGE

```
<?xml version = "1.0" standalone = "no" ?>
<!DOCTYPE CALLFOR_PROPOSAL_001 SYSTEM "9xx_callfor_proposal_001.dtd">
<PROPOSE_PROPOSAL_001>
      <CNTROLAREA>
            <BSR>
                  <VERB>PROPOSE</VERB>
                  <NOUN>PROPOSAL</NOUN>
                  <REVISION>001</REVISION>
            </BSR>
            <SENDER>
                  <LOGICALID>12345678</LOGICALID>
                  <COMPONENT>UF NEGOTIATION</COMPONENT>
                  <TASK>TASK001</TASK>
                  <REFERENCEID>112233</REFERENCEID>
                  <CONFIRMATION>1</CONFIRMATION>
                  <LANGUAGE>EN</LANGUAGE>
                  <CODEPAGE>TEST</CODEPAGE>
                  <AUTHID>ANY</AUTHID>
            </SENDER>
            <DATETIME QUALIFIER = "CREATION">
                  <YEAR>2000</YEAR>
                  <MONTH>01</MONTH>
                  <DAY>13</DAY>
                  <HOUR>10</HOUR>
                  <MINUTE>50</MINUTE>
                  <SECOND>19</SECOND>
                  <SUBSECOND>0874</SUBSECOND>
                  <TIMEZONE>-0500</TIMEZONE>
            </DATETIME>
      </CNTROLAREA>
   <DATAAREA>
         <PROPOSE_PROPOSAL>
            <NEGBODHEADER>
                  <PROPID>RFQ_10</PROPID>
                  <PRIMITIVE>PROPOSE</PRIMITIVE>
                  <SENDERNS>gator</SENDERNS>
                  <SENDERUID>SUP02</SENDERUID>
                  <RECEIVERNS>baghdad</RECEIVERNS>
                  <RECEIVERUID>BUY01</RECEIVERUID>
                  <NEGTRANSACTID>NS@baghdad2</NEGTRANSACTID>
                  <PROTOCOL>Bargaining</PROTOCOL>
            </NEGBODHEADER>
            <NEGITEM>
                  <ITEMID>computer_system</ITEMID>
                  <DESCRIPTN>computer</DESCRIPTN>
                  <USERAREA>userarea for computer_system</USERAREA>
                  <BUSINESSATTR>
                        <ATTRNAME>unit_price</ATTRNAME>
                        <ATTRTYPE>float</ATTRTYPE>
                        <CONSTRAINT>EQULA 1700.00 </CONSTRAINT>
                        <USERAREA>empty</USERAREA>
```

```
                    </BUSINESSATTR>
                    <BUSINESSATTR>
                          <ATTRNAME>quantity</ATTRNAME>
                          <ATTRTYPE>integer</ATTRTYPE>
                          <CONSTRAINT>RANGE[300..400]</CONSTRAINT>
                          <USERAREA>empty</USERAREA>
                    </BUSINESSATTR>
                    <BUSINESSATTR>
                          <ATTRNAME>delivery_day</ATTRNAME>
                          <ATTRTYPE>integer</ATTRTYPE>
                          <CONSTRAINT>RANGE[3..12]</CONSTRAINT>
                          <USERAREA>empty</USERAREA>
                    </BUSINESSATTR>
                    <ITEMATTR>
                          <ATTRNAME>model</ATTRNAME>
                          <ATTRTYPE>string</ATTRTYPE>
                          <CONSTRAINT>EQUAL "PentinumII 350"</CONSTRAINT>
                          <USERAREA/>
                    </ITEMATTR>
                          <ITEMATTR>
                          <ATTRNAME>monitor</ATTRNAME>
                          <ATTRTYPE>integer</ATTRTYPE>
                          <CONSTRAINT>ENUMERATION[15,17,19]</CONSTRAINT>
                          <USERAREA/>
                    </ITEMATTR>
                    <ITEMATTR>
                          <ATTRNAME>memory</ATTRNAME>
                          <ATTRTYPE>integer</ATTRTYPE>
                          <CONSTRAINT>RANGE[16m..64m]</CONSTRAINT>
                          <USERAREA/>
                    </ITEMATTR>
                    <ITEMATTR>
                          <ATTRNAME>hard_drive</ATTRNAME>
                          <ATTRTYPE>integer</ATTRTYPE>
                          <CONSTRAINT>RANGE[1g..12g]</CONSTRAINT>
                          <USERAREA/>
                    </ITEMATTR>
                    <INTERATTRCONSTR>
                          <CONSTRAINT>monitor = 15 implies unit_price <
                          1500.00</CONSTRAINT>
                          <USERAREA>userarea2</USERAREA>
                    </INTERATTRCONSTR>
              </NEGITEM>
        </PROPOSE_PROPOSAL>
    </DATAAREA>
</PROPOSE_PROPOSAL_001>
```

APPENDIX D
GENERATED NEGOTIATION STRATEGY RULE CODE IN JAVA

```
/* Database Systems Research and Development Center */
/* University of Florida */
/* copyright 1998 */

/* generated Java code for Rule : SR1  */
//import Utilities.*;
import java.io.*;
import java.util.Vector;
import CommonData.*;
public class SR1
{
      private String serPath;
       private String str_delivery_period;

       private String str_delivery_period_not_satisfied;

       private RuleResponse simRR;
      /* constructor  */
      public SR1()
      {
      }

      /* initialization  */
      public void init(String DirSer, String NameContext)
      {
            serPath = DirSer;
            str_delivery_period = new String("delivery_period");
                  str_delivery_period_not_satisfied = new
                  String("delivery_period can not be satisfied");

      }

      public boolean guard() {
            return true;
      }

      public boolean condition() {
            return  ((simRR.getLowBound(str_delivery_period)) > 10);
      }

      public Object  action() {
            simRR.downLowBound(str_delivery_period, 2);

            return simRR;     }
```

```
	public Object  altaction() {
		simRR.unResolvable(str_delivery_period_not_satisfied);

		return simRR;	}
	public java.lang.Object notifyEvent(Vector vRuleParam)
	{
		this.simRR = (RuleResponse)vRuleParam.elementAt(0);
		Object aRetVal = null;
		if (guard()){
			if (condition()){
				aRetVal = action();
			}
			else{
				aRetVal = altaction();
			}
		}
		return (java.lang.Object)aRetVal;
	}
}
```

LIST OF REFERENCES

[ALT99] AltaVisa UR: http://www.Altavista.com, 1999

[AMA99] Amazon.com URL: http://www.amazon.com, 1999

[AUC99] AuctionBot URL: http://auction.eecs.umich.edu, 1999

[BAR99] R. Bartak, "Constraint Programming: In Pursuit of the Holy Grail," in Proceedings of WDS99 (invited lecture), Prague, June 1999

[BEA96] C. Beam, A. Segev, and J.G. Shanthikumar, " Electronic Negotiation through Internet-based Auctions," *Technical Report*, University of California at Berkeley, 1996.

[BEA97a] C. Beam, A. Segev, "Automated Negotiation: A Survey of the Sate of the Art," *Technical Report*, University of California at Berkeley, 1997.

[BEA97b] C. Beam, A. Segev, "Electronic Catalogs and Negotiations," *Technical Report*, University of California at Berkeley, 1997.

[BIC98a] M. Bichler, A. Segev, "A Brokerage Framework for Internet Commerce", *Technical Report*, University of California at Berkeley, 1998.

[BIC98b] M. Bichler, A. Segev, J. Zhao, "Component-based E-Commerce: Assessment of Current Practices and Future Directions", *SIGMOD Records*, December, 1998, pp. 7 - 14

[BIC99] M. Bichler, M. Kaukal, A. Segev, "Mulit-attribute auctions for electronic procurement", *In Proceedings of the First Institute of Advanced Commerce (IBM) Workshop on Internet-based Negotiation Technologies*, New York, 1999

[BIN99] K. Binmore, N. Vulkan, "Applying game theory to automated negotiation", *Netnomics*, Vol. 1, No., 1, 1999

[BOL98] S. Boll, W. Klas, B. Battaglin, "*Design and Implementation of RMP - A Virtual Electronic Market Place* ", *SIGMOD Record*, December, 1998

[BRE91] J. W. Breslin, and J. Z. Rubin, "Negotiation Theory and Practice", *The Program on Negotiation at Harvard Law School*, Cambridge, MA, 1991.

[BRO85] L. Brownston, R. Ferrell, E. Kant, N. Martin, "Programming Expert Systems in OPS5: An introduction to rule-based programming", Addison-Wesley, Massachusetts, 1985

[BUI92] T. Bui, "Building DSS for Negotiators: A Three-Step Design Process." *In Proceedings of Hawaii International Conference on System Sciences,* Kauai, Hawaii, 1992

[CAT97] R.G.G. Cattell, "The Object database standard : ODMG 2.0" , Morgan Kaufmann, 1997.

[CHA94] S. Chakravarthy, E. Anwar, L. Maugis, and D. Mishra, "Design of Sentinel: an Object-Oriented DBMS with Event-Based Rules," Information and Software Technology, vol. 39(9), pp. 555--568, London, September, 1994.

[CHA96] A. Chaves, "An Agent marketpalce for buying and selling goods", *In proceedings of the First International Conference on the practical Application of Intelligent Agents and Multi-Agent Technology*, London,UK, 1996.

[CHA97] A. Chaves, "A Real-Life Experiment in Creating an Agent Marketplace." *In Proceedings of the Second International Conference on the practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, 1997.

[COL98] J. Collins, M. Tsvetovat, C. Bilot "A Market Architecture for Multi-Agent Contracting", *Second International Conference on AUTONOMOUS AGENTS*. 1998

[DAY88] U. Dayal, "The HiPAC Project: Combining Active Database and Time Constraints", ACM Sigmod Record, March, 1988

[DOG98] A. Dogac, I. Durusoy, S. Arpinar, N. Tatbul, P. Koskal, I. Cingil, N. Dimililer, "A Workflow-based Electronic Marketplace on the Web", *SIGMOD Record*, December, 1998

[DUJ81] J.J. Dujmovic, "ADM cost/benefit decision model: Mathematical models for data management system evaluation, comparison and selection", *Special Report 2 (Part 1),* National Bureau of Standards, contract NB80SBCA0449, July 1981. Available from NTIS: PB82-170150

[DWO96] G. Dworman, S. Kimbrough, J. Laing, "On Automated Discovery of Models Using Genetic Programming in Game-Theoretic Contexts," *Journal of Management Information Systems*, vol. 12(3), Winter 1996.

[EBA99] Ebay URL: http://www.ebay.com, 1999

[FAR96] A. Farquhar, R. Fikes, & J. Rice. "The Ontolingua Server: A Tool for Collaborative Ontology Construction", *Technical Report*, Knowledge Systems Laboratory, Stanford University, KSL-96-26, September 1996

[FOX94] M. Fox and M. Gruninger, "Ontologies for Enterprise Integration," in *Proceedings of the Second International Conference on Cooperative Information Systems (CoopIS)*, Toronto, Canada, pp. 82-89, 1994.

[FIK68] R. Fikes, "A Heuristic Program for Solving Problems Stated as Non-deterministic Procedures", Ph.D. thesis, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA, 1968

[FIS91] R. Fisher, U. William, B. Patton, "Getting to Yes: Negotiating Agreement Without Giving In", 2nd ed. Boston, Houghton Mifflin, 1991.

[FIP97] FIPA 97 Specification Version 2.0, Part 2, "Agent Communication Language", 1997, URL: http://www.fipa.org/spec/fipa97.html, 1997

[FIS99] Fishmarket URL: www.iiia.csic.es/Projects/fishmarket/newindex.html, 1999

[GAS74] J. Gaschnig, "A Constraint Satisfaction Method for Inference Making", In Proceedings of 12th Annual Allterton Conference on Circuit System Theory, pp. 866-874, 1974

[GEH91] N. Gehani, H. Jagadish "ODE as an Active Database: Constraints and Triggers," in Proceedings of the 17th. VLDB Conference, 1991.

[GIA91] J. Giarratano, "CLIPS User's Guide, Volume 1, Rules", NASA, Lyndon B. Johnson Space Center, Information Systems Directorate, Software Technology Branch, Houston, Texas, 1991

[GOL65] S. Golomb, L. Baumert, "Backtrack Programming", Journal of Association for Computing Machinery (ACM), vol 12, pp.516-524, 1965

[GRO99] B. Grosof, Y. Labrou, "An Approach to using XML and a Rule-based Content Language with an Agent Communication Language," In *Proceedings of the IJCAI-99 Workshop on Agent Communication Languages (ACL-99),* Stockholm, Sweden, Aug. 1, 1999.

[GUT98] R. Guttman, P. Maes, "Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce", *Proceedings of the Second International Workshop on Cooperative Information Agents (CIA'98).* Paris, France, July 1998

[HAL92] F. Halsall, "Data Communications, Computer Networks, and Open Systems", Addison-Wesley, 1992

[HAN96] Eric N. Hanson, "The Design and Implementation of the Ariel Active Database Rule System," IEEE Trans. on Knowledge and Data Engineering, vol. 8, no. 1, February 1996.

[HAS90] L. Haas, W. Chang, G. Lohman, J. McPherson, P.F. Wilms, G. Lapis, B. Lindsay, H. Pirahesh, M. Carey, and E. Shekita, "Starburst mid-flight: As the dust clears," IEEE Transactions on Knowledge and Data Engineering, vol. 2(1):143-160, March 1990.

[JEL89] M. Jelassi, and A. Foroughi, "Negotiation Support Systems: An Overview of Design Issues and Existing Software." *Decision Support Systems*, 1989. 5(2): p. 167-181.

[JON80] A. Jones, "Game Theory: Mathematical Models of Conflict", Ellis Horwood Ltd., Chichester, England, 1980

[KAR97] N. Karacapilidis, C. And Pappis, "A framework for group decision support systems: Combining AI tools and OR techniques", *In European Journal of Operational Research*, No 103, 1997

[KAS99] Kasbah URL: http://kasbah.media.mit.edu, 1999

[KEE76] R. Keeney, H. Raifaa, "Decisions with Multiple Objective: Preference and Value Tradeoffs", John Wiley & Sons, 1976

[KEL95] A. Keller, "Smart Catalogs and Virtual Catalogs," *In International Conference on Frontiers of Electronic Commerce*, October 1995,

[KER99] G.E. Kersten, S. Szpakowicz "Modelling Business Negotiations for Electronic Commerce", technical report, International Institute for Applied Systems Analysis, Austria, 1999

[KOI98] J. Koistinen, A. Seetharaman, "Worth-Based Multi-Category Quality-of-Service Negotiation in Distributed Object Infrastructures", HP Technical Report, 1998

[KUM98] M. Kumar, S. I. Feldman, "Business negotiation on the Internet." IBM Institute for Advanced Commerce (IAC) Technical Report, 1998

[LAM98] H. Lam and S.Y.W. Su, "Component Interoperability in a Virtual Enterprise Using Events/Triggers/Rules", In *Proc. of OOPSLA '98 Workshop on Objects, Components, and Virtual Enterprise*, Oct. 18-22, 1998, Vancouver, BC, Canada.

[LEE97] J. G. Lee, "ICOMA: An open infrastructure for agent-based intelligent Electronic Commerce on the Internet ", In *Proceedings of the*

International Conference on Parallel and Distributed Systems (CPADS). IEEE Comp Soc, Los Alamitos, CA, pp 648-655, 1997.

[LEE99] M. Lee, "Event and Rule Services for Achieving Internet-based Knowledge Networks", Ph.D. Proposal, University of Florida, 1999.

[LEW97] R. Lewicki, "Essentials of Negotiation", Irwin, 1997

[LIM93] L. Lim, I. Benbasat, "A Theoretical Perspective of Negotiation Support Systems", *in Journal of Management Information Systems*, Vol. 9, No. 3, 1993

[LIN92] P. Linhart, R. Radner, M. Satterthwaite, "Bargaining with Incomplete Information", Academic Press, San Diego, 1992

[MAT91] S. Matwin, T. Szapiro, K. Haigh, "Genetic Algorithms Approach to a Negotiation Support System." *IEEE Transactions on Systems, Man, and Cybernetics,* Vol. 21, p. 102-114, 1991

[MEL98] B. Meltzer, R. Glushko, "XML and Electronic Commerce: Enabling the Network Economy ", *SIGMOD Record,* December, 1998

[MOU98] A. Moukas, R. Guttman, P. Maes. "Agent-mediated Electronic Commerce: An MIT Media Laboratory Perspective." *Proceedings of the First International Conference on Electronic Commerce (ICEC'98),* Seoul, Korea, April 1998

[MUL96] H. J. Muller, "Negotiation Principles", Chapter 7, *Foundations of Distributed Artificial Intelligence*, John Wiley & Sons, Inc, 1996.

[MYE83] R. Myerson, M. Satterthwaite, "Efficient Mechanisms for Bilateral Trading." *Journal of Economic Theory* Vol. 29, pp. 265-81, 1983

[NAS50] J. Nash, "The Bargaining Problem", *Econometrica*, Vol. 18, pp. 155-162, 1950

[NAS53] J. Nash, "Two Person Cooperative Games", Econometrica, Vol. 21, p. 128-140, 1953

[OAG99] Open Application Group URL: http://www.oag.org, 1999

[OBJ92] Object Management Group, "The common Object Request Broker: Architecture and Specification", John Wiley & Sons, Inc. New York, 1992

[OHR96] G. O'Hare, N. Jennings, "Foundations of Distributed Artificial Intelligence", John Wiley & Sons, Inc., New York, NY, 1996

[OLI97] J. Oliver, "A Machine Learning Approach to Automated Negotiation and Prospects for Electronic Commerce", *Journal of Management Information Systems*, Vol. 13, pp. 83-112, 1997

[ONS99] Onsale URL: http://www.onsale.com, 1999

[OSG62] C. Osgood, "An alternative to war or surrender". University of Illinois Press, Urbana, IL. 1962.

[PRU81] D. Pruitt, "Negotiation Behavior", Academic Press, 1981.

[RAI82] H. Raiffa, "The Art and Science of Negotiation", Harvard University Press, Cambridge, MA, 1982.

[RAN97] A. Rangaswamy, G. Shell, "Using Computers to Realize Joint Gains in Negotiations: Toward an Electronic Bargaining Table", in *Management Science*, Vol. 43, No. 8, August 1997

[REE99] D. Reeves, M. Wellman, B. Grosfot, H. Chan, "Automated Negotiations from Formal Contract Descriptions", *In Proceedings of the First Institute of Advanced Commerce (IBM) Workshop on Internet-based Negotiation Technologies*, New York, 1999

[REI98] B. Reich, I. Ben-Shaul, "A Componentized Architecture for Dynamic Electronic Markets", *SIGMOD Record*, December, 1998

[ROD97] J. Rodrquesz, "FM96.5: A Java-based Electronic Auction House." *Processings of the Second International Conference on the Pratical Application of Intelligent Agents and Multi-Agent Technology.* London, UK, April 1997

[ROS94] J. Rosenschein, G. Zlotkin, "Rules of Encounter: Designing Conventions for Automated Negotiation among Computers", MIT press, Cambridge, MA, 1994

[ROT79] A. Roth, "Axiomatic Models of Bargaining", Berlin/New York, Springer, 1979

[SAN95] T. Sandholm V. Lesser., "Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework", In *First International Conference on Multi-Agent Systems (ICMAS-95)*, San Francisco, CA, 1995.

[SEG95] A. Segev, D. Wan, C. Beam, "Designing Electronic Catalogs for Business Value: Results from the Commerce Net Pilot," Technical Report, CITM-95-WP-1005, Fisher Center for Information Technology and Management, University of California at Berkeley, October 1995,

[SIE97] C. Sierra, P. Faratin, N. Jennings: "A Service-Oriented Negotiation Model between Autonomous Agents, In *Proceedings of 8th European Workshop on Modeling Autonomous Agents in a Multi-Agent World (MAAMAW-97)*, Ronneby, Sweden, 17-35, 1997

[SMI80] R. Smith, "The contract net protocol: High-level communication and control in a distributed problem solver." In *Readings in Distributed Artificial Intelligence* (A. Bond and L. Gasser, eds.), Morgan Kaufmann, San Mateo, CA, 1980.

[SOK89] P. Sokol. "EDI: The Competitive Edge", Intertext Publications, McGraw-Hill Book Company, New York, Y, 1989

[SU93] S.Y.W. Su, H. Lam, S. Eddula, J. Arroyo, N. Prasad, R. Zhuang, "OSAM*.KBMS: An Object-oriented Knowledge Base Management System for Supporting Advanced Applications," Proceedings of ACM SIGMOD, Washington D.C., pp. 540-541, May 25-28, 1993

[SU87] S. Y. W. Su, J. Dujmovic, D. S. Batory, S. B. Navathe, and R. Elnicki, "A Cost-Benefit Decision Model: Analysis, Comparison, and Selection of Database Management Systems," *ACM Transactions on Database Systems*, Vol. 12, No. 3, Sept. 1987, 472-520.

[TAT96] A. Tate, "Towards a Plan Ontology", Technical report, Artificial Intelligence Application Institute, The University of Edinburgh, Unite Kingdom, 1996

[TSA93] E. Tsang, "Foundation of Constraint Satsifaction", Academic Press, New York, 1993

[TYG98] J. Tygar, "Atomicity versus Anonymity: Distributed Transactions for Electronic Commerce", *Proceedings of the 24th Annual International Conference on VLDB*, 1998, pp. 1- 12

[W3C98] Word Wide Web Consortium, "eXtensib Markup Lanauge", http://www.w3.org/XML, 1998

[WAL72] D. Waltz, "The Constraint Language Trilogy: Semantics and Computations," Technical Report, Complete Logic Systems, North Vancouver, Canada, 1972

[WAL96] R. Wallace, E. Freuder, "Heuristic Methods for Over-Constrained Constraint Satisfaction problems", In Proceedings of International Conference on Principles and Practice of Constraint Programming, Cambridge, Massachusetts, 1996

[WID96] J. Widom, S. Ceri, "Active database systems: triggers and rules for advanced database processing", Morgan Kaufmann, 1996.

[WOO90] C. Woo, M. Chang, "An Approach to Facilitate the Automation of Semistructured and Recurring Negotiations in Organizations.", *Journal of Organizational Computing*, Vol. 2(1), p. 47-76, 1990

[WUR98] P. Wurman, M. Wellman. "The Michigan Internet AuctionBot: A configurable Auction Server for Human and Software Agents." *Proceedings of the Second International conference on Autonomous Agents*, May 1998

[WUR99] P. Wurman, M. Wellman, "A Control Architecture for Flexible Internet Auction Servers," *In Proceedings of the First Institute of Advanced Commerce (IBM) Workshop on Internet-based Negotiation Technologies*, New York, 1999

[YAH99] Yahoo URL: http://auctions.yahoo.com, 1999

[ZEN98] D. Zeng, K. Sycara, "Bayesian Leaning in Negotiation," *International Journal of Human Computer Systems*, Vol. 48, pp. 125-141, 1998

[ZLO96] G. Zlotkin, S. Jeffrey, A. Rosenschein, "Mechanism design for automated negotiation and its application to task oriented domains", *Artificial Intelligence*, Vol. 86, pp. 195- 244, 1996

BIOGRAPHICAL SKETCH

Chunbo Huang was born in Quxian, Sichuan Province, P. R. China, in 1972. He received his Bachelor of Engineering degree in Computer Science and Technology in 1994 and Master of Engineering degree in Computer Science and Technology in 1996, both from Tsinghua University, Beijing, P. R. China. From 1996 to Present, Chunbo Huang has been a Ph.D. Student in Computer and Information Science and Engineering Department at the University of Florida. He also works as a research assistant in the Database System Research and Development Center under the guidance of Professor Stanley Y. W. Su. Chunbo Huang's primary research interests include the automated negotiation, object-oriented databases, active databases, system integration, electronic commerce/business system, and supply chain management system.

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Stanley Y.W. Su, Chairman
Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Joachim Hammer, Co-chairman
Assistant Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Herman Lam
Associate Professor of Electrical and Computer Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Abdelsalam Helal
Associate Professor of Computer and Information Science and Engineering

I certify that I have read this study and that in my opinion it conforms to acceptable standards of scholarly presentation and is fully adequate, in scope and quality, as a dissertation for the degree of Doctor of Philosophy.

Antonio A. Arroyo
Associate Professor of Electrical and Computer Engineering

This dissertation was submitted to the Graduate Faculty of the College of Engineering and to the Graduate School and was accepted as partial fulfillment of the requirements for the degree of Doctor of Philosophy.

May 2000

M.J. Ohanian
Dean, College of Engineering

Winfred M. Phillips
Dean, Graduate School